Universidade de Aveiro Departamento de Engenharia Mecânica
2017

Pedro Cruz Duarte

Sistema *front-end* de máquina de ensaios

Universidade de Aveiro Departamento de Engenharia Mecânica
2017

Pedro Cruz Duarte

Sistema *front-end* de máquina de ensaios

Dissertação apresentada à Universidade de Aveiro para cumprimento dos requisitos necessários à obtenção do grau de Mestrado em Engenharia Mecânica, realizada sob orientação científica de Rui António da Silva Moreira, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro da Universidade de Aveiro e de José Paulo Oliveira Santos, Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro da Universidade de Aveiro.

Apoio financeiro dos projetos UID/EMS/00481/2013-FCT e CENTRO-01-0145-FEDER-022083

O júri

Presidente	**Prof. Doutor Miguel Armando Riem de Oliveira** Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro
Arguente	**Prof. Doutor Pedro Nicolau Faria da Fonseca** Professor Auxiliar do Departamento de Eletrónica, Telecomunicações e Informática da Universidade de Aveiro
Co-Orientador	**Prof. Doutor José Paulo Oliveira Santos** Professor Auxiliar do Departamento de Engenharia Mecânica da Universidade de Aveiro

Agradecimentos

Ao professor Rui Moreira, pelo esforço de acompanhamento do trabalho apesar da lesão sofrida este ano e da impossibilidade de presença física na Universidade, por me proporcionar o equipamento necessário e as condições para o seu desenvolvimento. Desejo as melhoras rápidas.

Ao professor José Paulo Santos, pela ampla disponibilidade demonstrada no esclarecimento de dúvidas, pela vontade de ajuda e interesse demonstrados.

Aos meus colegas de laboratório, pela ajuda prestada na análise do meu trabalho, pela discussão construtiva de diferentes soluções, pela introdução e ambientação a novos conteúdos e pela camaradagem que tornou a estada no laboratório para o desenvolvimento da máquina mais aprazível e estimulante.

Aos meus familiares e amigos, pelo apoio constante e motivador durante esta fase da minha vida, pressionando e acelerando o processo de desenvolvimento de trabalho, lembrando-me do objetivo pessoal que é assim concluído.

Palavras-chave Máquina de ensaios materiais; Sistema *front end*; Interface gráfico do utilizador

Resumo O trabalho elaborado consiste no desenvolvimento de um sistema de *front end* de uma máquina de ensaios elaborada no laboratório de dinâmica de estruturas da universidade. A máquina de ensaios é baseada num atuador elétrico linear com um encoder de alta resolução e conta com um transdutor de força analógico. Destes últimos dois equipamentos são retirados os dados de ensaio através do uso de um *Arduino* e uma *shield* desenvolvida para o efeito. A associação dos equipamentos de atuação e leitura de dados é feita através de um interface gráfico desenvolvido para uma plataforma *Raspberry Pi*. O sistema de *front end* desenvolvido simplifica a utilização da máquina na recolha de dados e programação de testes. Através das alterações implementadas na máquina de ensaios, conseguiu-se uma solução mais intuitiva, rápida, segura e fácil de utilizar que a solução anterior.

Keywords Material testing machine; Front end system; Graphical user interface

Abstract The elaborated work consists in the development of a front end system of a test machine developed in the universitys structure dynamics laboratory. The testing machine is based on a linear electric actuator with a high resolution encoder and has an analog power transducer. The test data are obtained of these last two equipments through the use of an Arduino and a shield developed for the purpose. The combination of the equipment of application of power and data reading is done through a graphic interface developed for a platform Raspberry Pi. The developed front end system simplifies the machine usage in data recollection and test programming. Thru the implemented alterations in the testing machine a more intuitive, fast, secure and easy to use solution was achieved.

Índice

Lista de Tabelas

Lista de Figuras

Capítulo 1

Introdução

1.1 Enquadramento

No intuito de investigação de novos materiais para a indústria biomédica e de biomateriais, em parceria com outras instituições, foi desenvolvida uma máquina de ensaios em 2009, cujo objetivo era testar o comportamento cíclico de reforços para tecidos ligamentosos com fibras compósitas biodegradáveis de base polimérica. Este estudo tinha como propósito auxiliar a recuperação de lesões ou cirurgia através da transferência de carga para o reforço estudado, sendo este bio compósito degradado através da sua utilização cíclica, desaparecendo do sistema após a reabilitação do ligamento.

A máquina de ensaios foi desenvolvida com a finalidade de testar e otimizar o comportamento do material, sendo os parâmetros principais a observar a deformação máxima aplicada e o número de ciclos antes da cedência. Para a análise desses parâmetros foi implementada na máquina um método de atuação e de leitura de dados de força e deslocamento funcional, mas passível de melhorias de precisão e estabilidade.

Após o projeto inicial, a máquina antes desenvolvida foi utilizada noutros trabalhos com o intuito de desenvolvimento, teste e comparação de outros materiais e biomateriais.

1.2 Motivação e Objetivos

Este projeto foi proposto com o objetivo de melhorar o sistema existente, tornando a utilização da máquina em futuros projetos, investigações ou teses mais viável e eficiente.

Para tal foi proposto o desenvolvimento de um sistema baseado em Arduino para controlo e interface com o utilizador de uma máquina de ensaios desenvolvida no laboratório de dinâmica de estruturas do departamento de Engenharia Mecânica da Universidade de Aveiro.

A máquina de ensaios é baseada num atuador elétrico linear com encoder de alta resolução e conta com um transdutor de força analógico. Existe já uma solução Arduino para interface do encoder e do transdutor de força, que se pretende ser melhorada através da utilização de um novo integrado de descodificação de encoder e um condicionador de transdutor de força, para além de uma interface html.

O front end de interface com o utilizador deverá permitir uma ligação direta à plataforma Arduino. Pretende-se que o sistema desenvolvido venha simplificar a interação do utilizador com a máquina, tornando-a numa ferramenta mais útil e requerida na exerção de investigação na área de propriedades de materiais. A solução desenvolvida deverá aglomerar os equipamentos referidos e outros que possam ser necessários, permitindo ao mesmo tempo flexibilidade para troca dos mesmos, resultando assim num sistema modular. A futura troca dos módulos é necessária para facilitar futuras melhorias pontuais do sistema ou utilização noutras aplicações.

Deverá também ser aumentada a segurança e resposta a emergência do sistema, estando o sistema corrente apenas dotado de seguranças internas aos equipamentos de alimentação e controlo do atuador.

1.3 Organização do documento

Este projeto encontra-se organizado em 5 capítulos, cuja separação tem como intuito facilitar a compreensão do trabalho realizado, reunindo sobre o mesmo tema as informações necessárias para compreender cada etapa do projeto da maneira mais coesa possível. Assim sendo, os seguintes capítulos vão-se dedicar ao estudo do corrente projeto nos seguintes temas:

- Capítulo 1:

  O primeiro capítulo do trabalho serve para apresentar o tema e a estrutura do documento, focando-se também nos motivos que desencadearam a necessidade de realização do projeto e nos objetivos que pretende solucionar;

- Capítulo 2:

  Neste capítulo é realizada uma apresentação dos conceitos necessários para a compreensão deste trabalho, sendo estes também úteis para a contextualização do desenvolvimento de uma máquina de ensaios. Assim sendo, serão analisados conceitos aliados ao tema de testes materiais, assim como será realizada uma análise de estado de arte dos equipamentos basilares;

- Capítulo 3:

  Neste capítulo pretende-se realizar uma análise incisiva do funcionamento da máquina de ensaios anterior a qualquer modificação inserida neste projeto. Esta análise foca-se no modo de funcionamento da mesma, nos equipamentos utilizados e na sua interação, pretendendo auxiliar a reforma da mesma de maneira mais eficaz e produtiva e fundamentar os objetivos delineados para o projeto;

- Capítulo 4:

  À semelhança com o capítulo anterior, neste capítulo refaz-se uma análise do funcionamento da máquina de ensaios, agora posteriormente à intervenção realizada no âmbito deste projeto, focando-se nos mesmos pontos observados na análise prévia. Este capítulo tem como intuito apresentar o trabalho realizado, assim como justificar as decisões que o seu desenvolvimento acarreta;

- Capítulo 5:

  Por fim, é sumariada a comparação das diferentes soluções, concluindo-se sobre a conformidade do trabalho realizado com os objetivos iniciais do projeto. Neste capítulo são também realizadas algumas propostas para desenvolvimento de soluções semelhantes.

Capítulo 2

Revisão Bibliográfica

2.1 Tipo de ensaios e provete para teste material

Cada material possui propriedades únicas e intrínsecas que o caracterizam. Estas propriedades definem as condições em que podem ser utilizados, sendo de maior importância para um engenheiro e projetista conhecê-las, de maneira a evitar falhas na utilização ou sobre-dimensionamento do produto.

Do ponto de vista de projeto mecânico, um dos aspetos mais importantes na escolha do material a utilizar são as suas propriedades mecânicas, ou seja, a capacidade de reação a esforços de natureza mecânica, afetando não só a aplicabilidade do material como os processos de fabrico do produto final. Noutras aplicações as características mecânicas também podem desempenhar um papel importante, mesmo quando a função requerida é de natureza elétrica, magnética, ótica ou biológica (15).

Apesar de nenhum ensaio mecânico poder prever completamente o comportamento real de um material enquanto executa a sua função, estes permitem obter informação sobre as suas propriedades, possibilitando uma aproximação do comportamento expectável. Estas propriedades dos materiais são verificadas e estabelecidas através da realização de diversos tipos de ensaios, permitindo recolher informação para determinar o seu comportamento em situações padrão, como a aplicação de cargas estáticas ou variáveis, alterando os seus fatores como a magnitude, natureza e duração da carga, bem como as condições ambientais. O comportamento mecânico do material reflete a relação entre a sua resposta ou deformação a uma tensão ou força aplicada (1). A tensão pode apresentar diversos perfis, como de tração, compressão e de corte, como se pode observar na figura 2.1 , mas no geral é definida como uma força atuante numa área.

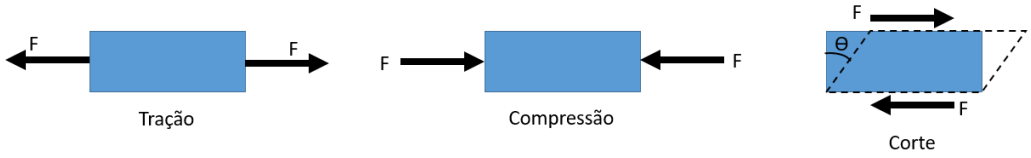

Figura 2.1: Força de tração, compressão e corte, respetivamente na imagem

A deformação ocorre quando se aplica uma carga ao material. Ao aplicar uma força de tração uniaxial num varão, este alonga-se segundo a direção de aplicação da força. "Esse deslocamento é designado por deformação. Por definição, a extensão nominal que é provocada pela ação da força de tração uniaxial aplicada à amostra metálica é dada pelo quociente entre a variação do comprimento da amostra segundo a direção de aplicação da força e o comprimento inicial da amostra." (16) (Equação 2.1). A deformação pode ser plástica ou elástica, sendo considerada plástica quando, após retirada a carga, a forma do provete de teste se altera em relação à inicial, e elástica quando volta à sua forma inicial.

$$\varepsilon = \frac{l_i - l_0}{l_0} = \frac{\Delta l}{l_0} \tag{2.1}$$

Por uma questão prática, muitas vezes por não ser possível fazer os testes com o produto final em situações expectáveis na sua vida útil e por muitos dos testes serem possivelmente destrutivos ou danosos, são usados provetes de diversas geometrias em testes padrão para obter valores de propriedades relevantes para o seu uso, a partir dos quais se pode simular ou depreender o comportamento do material em uso ou fabrico. De seguida vão-se analisar diferentes tipos de ensaios relevantes para o presente trabalho e explicar que propriedades se pode obter a partir dos mesmos.

2.1.1 Ensaio de tração

Máquinas de ensaios de tração aplicam uma carga uniaxial de uma maneira uniforme e são geralmente universais nas suas capacidades e aplicações, em vez de serem específicas para um tipo de teste ou material. O efeito da instrumentação nos resultados dos testes deve ser minimizado para evitar erros de teste. A gravação automática da tensão e extensão é altamente desejável, e módulos que possibilitam funções especificas, como desligar automático quando o espécime quebra, simplificam o uso da máquina (17).

O ensaio de tração é utilizado para avaliar a resistência mecânica de metais e ligas. Neste ensaio, traciona-se um provete do material até à fratura, num intervalo de tempo relativamente curto e com uma velocidade constante. Os valores da força obtidos podem ser convertidos em valores da tensão nominal, o que permite construir um gráfico da tensão nominal em função da extensão nominal (16), como o que pode ser observado na Figura 2.2.

Este ensaio permite extrair informação sobre o módulo de elasticidade, recorrendo à Equação 2.2 ou calculando o declive da reta observável na Figura 2.2 na zona elástica, desde o inicio do teste até à tensão de cedência, "Tensão a partir da qual a deformação plástica do metal ou liga metálica passa a ser significativa." (16).

$$E = \frac{\sigma(\text{Tensão})}{\varepsilon(\text{Extensão})} \tag{2.2}$$

A tensão de rutura, tensão máxima ou resistência à tração pode ser também retirado do gráfico presente na Figura 2.2 através do ponto de maior ordenada, ou seja, o ponto de maior força exercida, assumindo-se muitas vezes uma área de secção de provete ainda igual à inicial. Este ponto marca o fim da deformação uniforme e o início de uma deformação localizada, originando uma diminuição na área da secção transversal do provete

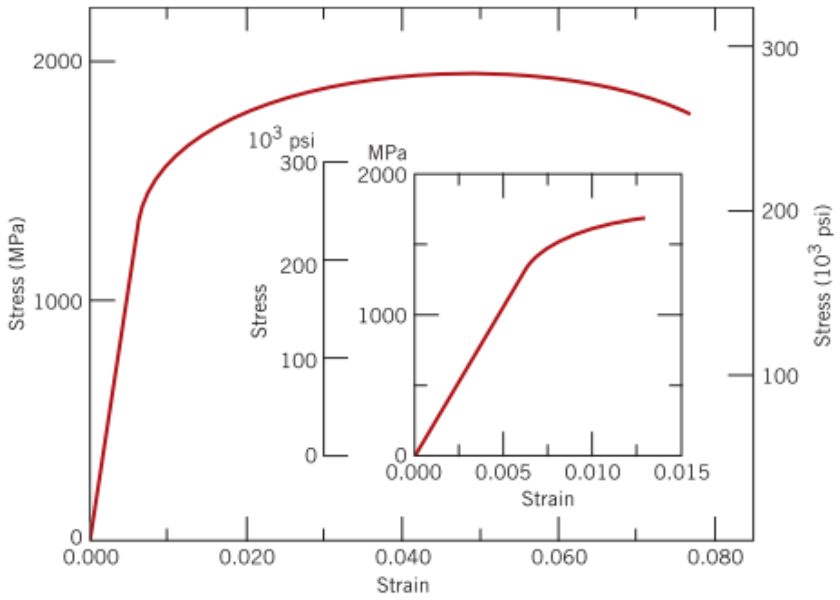

Figura 2.2: Gráfico Tensão-Extensão para uma liga de aço (1)

mais acentuada. Após a rotura do provete, marcado no gráfico da Figura 2.2 como o último ponto da curva, é possível retirar informação sobre o alongamento percentual até à fratura, que é o alongamento que um provete sofre durante o ensaio e fornece um valor para a ductilidade do material. (16) e coeficiente de estrição, que representa a variação máxima da área da secção transversal medida após a rotura do provete, expressa em percentagem da área da secção inicial da zona útil (18).

2.1.2 Ensaio de fluência

"Um metal ou liga metálica submetido a uma carga ou tensão constante pode sofrer uma deformação ao longo do tempo. Esta deformação ao longo do tempo designa-se por fluência" (16).

A curva de fluência, é determinada através de um ensaio realizado a temperatura e carga constante e em que a deformação do provete é registada ao longo do tempo. A duração dos ensaios depende, entre outros fatores da carga/tensão e da temperatura. Durações de pelo menos 2000 horas (84 dias) são frequentes, podendo contudo atingir os vários meses ou mesmo anos (18).

É frequente realizarem-se múltiplos ensaios, com diversos níveis de tensão e temperatura, de forma a abranger uma maior gama de condições de aplicação. Em ensaios de fluência com rutura é normal traçar-se o gráfico tensão-tempo (Figura 2.3) até rutura em escala logarítmica. Este ensaio permite compreender o comportamento do material em utilizações de carga estática e prever uma vida útil para peças cuja tolerância dimensional pode ser crítica para o funcionamento do sistema (18).

No início do teste, devido à aplicação da carga, pode existir uma deformação inicial, podendo ser plástica se a tensão for elevada o suficiente. Após esse instante identificam-se três zonas distintas na curva: Zona de fluência primária, secundária e terciária, iden-

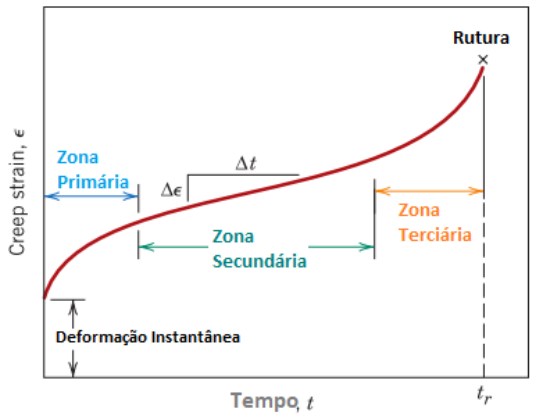

Figura 2.3: Curva de fluência típica a uma temperatura elevada. (1)

tificadas na Figura 2.3. Na zona de fluência primária verifica-se uma diminuição da velocidade de fluência com o tempo (Equação 2.3). Na zona seguinte, designada por zona de fluência secundária, a velocidade de fluência mantém-se estável e praticamente constante, devido ao equilíbrio entre os efeitos contrários do encruamento e dos mecanismos de libertação de deslocações. Ao valor médio da velocidade de fluência durante este período designa-se normalmente por velocidade de fluência mínima (Equação 2.3), e é um dos parâmetros retirados deste teste com mais importância na escolha de materiais.

$$\text{Velocidade de Fluência} = \frac{\Delta\varepsilon}{\Delta t} \tag{2.3}$$

A zona de fluência terciária normalmente só se verifica em testes de elevada tensão e/ou temperatura, sendo o resultado de instabilidade estrutural e mecânica. Devido a estas instabilidades, originam-se zonas de acumulação de tensão que poderão provocar uma diminuição de área da secção transversal e aumentar a velocidade de deformação.

2.1.3 Ensaio de flexão

O ensaio de flexão consiste em sujeitar um provete a uma deformação plástica com a finalidade de avaliar de forma comparativa a ductilidade dos materiais e detetar defeitos de compacidade e heterogeneidades do material (18). Existem vários tipos de ensaios por flexão, entre os quais a flexão com três ou quatro pontos, que consiste num punção com dois apoios; dobragem por flexão, onde se força o material a dobrar em torno de um punção fixo com uma das extremidades presa ao mesmo; e ensaio com punção e matriz, onde o provete é dobrado até entrar em contacto com as paredes da matriz.

Apesar de não fornecerem resultados quantitativos, estes tipos de ensaio são um método eficaz, embora destrutivo, de detetar defeitos no material, como inclusões e problemas de compacidade, sendo utilizados primariamente no controlo de qualidade de produtos fabricados (17, 18). Este tipo de teste permite ainda verificar a capacidade de dobragem e enformação dos materiais.

2.1.4 Ensaio de fadiga

"A falha por fadiga resulta da iniciação de uma microfenda e subsequente propagação sob solicitações cíclicas" (18). Nestas circunstâncias é possível que a falha ocorra a um nível de tensão consideravelmente inferior à tensão de cedência ou rutura do material em ensaios estáticos. O termo "fadiga" é empregue porque este tipo de falha ocorre geralmente após um longo período de tempo de repetição de cargas ou deformações (1). O ensaio de fadiga consiste na aplicação de uma carga variável cíclica ou aleatória no provete, e permite o estudo do comportamento do material a diferentes ciclos de carga, diferentes frequências de atuação e a diferentes números de ciclos. A carga pode ser axial (tração ou compressão), de flexão (dobragem) ou de torção de natureza (1). "Desde os estudos de Wohler que uma das formas mais comuns de apresentação de resultados de fadiga, é o traçado das chamadas curvas de Wohler ou S-N, em gráficos de tensão (S) versus o número de ciclos (N) até à falha do provete. A tensão a considerar poderá ser σ_{max}, σ_a ou a σ_g" (18), ou seja, a tensão máxima, a amplitude de tensão (Equação 2.4) ou a gama de tensão (Equação 2.5).

$$\text{Amplitude de tensão } (\sigma) = \frac{\sigma_{max} - \sigma_{min}}{2} \tag{2.4}$$

$$\text{Gama de tensão } (\sigma_g) = \sigma_{max} - \sigma_{min} \tag{2.5}$$

Este ensaio é relevante também para analisar a vida útil de um componente em uso, uma vez que a fadiga é uma das maiores causas de falha em componentes ou estruturas (18). É estimado que cerca de 90% de todas as falhas de metais seja por fadiga, e é particularmente importante por ser catastrófica e acontecer subitamente e muitas vezes sem sinais aparentes (1).

2.2 Tipo de *hardware* para testes materiais

Para a realização dos ensaios pretendidos é preciso analisar o *hardware* necessário para o seu correto desempenho. Para tal vão-se de seguida analisar as principais opções no mercado para os três componentes principais de qualquer máquina de teste: o método de atuação, isto é, a maneira como se vai gerar a potência para aplicar a tensão no provete; o modo de ler o carregamento aplicado e o processo de medição de deformação provocada pelo carregamento aplicado.

2.2.1 Forma de atuação

A aplicação de potência para a realização de ensaios materiais é um problema com bastantes soluções de mercado, como a atuação pneumática, hidráulica e elétrica, pelo que a escolha requer o conhecimento das vantagens e desvantagens de cada tipo. A escolha do atuador depende da função principal a realizar e dos seus parâmetros base, como a força máxima permitida, a sua velocidade de atuação, o percurso máximo requerido, entre outros.

Um atuador é um nome genérico que se refere a um dispositivo que converte uma energia de entrada em energia mecânica, e vários atuadores foram desenvolvidos e postos em prática de acordo com várias entradas de energia (3).

A maioria dos processos industriais requer o movimento de objetos ou matérias primas de uma localização para outra, ou a aplicação de uma força para prender, formar ou comprimir um produto. Em muitos desses processos são utilizados atuadores elétricos (19).

Um sistema elétrico proporciona várias escolhas básicas: o solenoide, o motor CC[1], o motor de indução CA[2] ou o motor de passo. Destes, o solenoide proporciona diretamente um deslocamento linear, enquanto as outras opções precisam de conversão do movimento rotativo que produzem para o movimento linear, existindo diversas soluções comerciais para a resolução desse obstáculo presentes no mercado. Um motor de corrente contínua apresenta um controlo de velocidade superior, mas têm mais necessidade de manutenção, enquanto que os motores de corrente alternada apresentam características inversas, sendo necessário o controlo da frequência da corrente para controlar a velocidade de rotação do mesmo, mas não apresentam virtualmente nenhuma necessidade de manutenção.

Todos os motores elétricos são governados pelas leis do eletromagnetismo, e estão sujeitos essencialmente às mesmas limitações impostas pelos materiais que os compõem (cobre e ferro), pelo que é natural que o seu funcionamento básico seja bastante semelhante (2).

Num motor de corrente continua, a rotação é provocada pela interação entre a corrente elétrica que percorre os condutores axiais do rotor e o fluxo magnético radial produzido pelo estator. O fluxo ou excitação pode ser desenvolvido por ímanes permanentes ou bobines. O circuito principal consiste num par de bobinas enroladas em ranhuras no rotor, formando a armadura. Corrente é fornecida a estas bobinas através de escovas de carbono que fazem contacto deslizante com o comutador, que consiste em segmentos de cobre isolados montados de maneira a formar um cilindro que gira com o rotor. Este comutador tem como função ativar as espiras necessárias para estabelecer o fluxo correto de corrente enquanto o motor gira. Como se pode observar na Figura 2.4, os condutores mais próximos do polo N conduzem corrente num sentido, suportando assim uma força perpendicular à corrente e ao fluxo magnético, enquanto que os mais próximos do polo S conduzem no sentido oposto, sopurtando uma força inversa proporcional à densidade de fluxo magnético axial e à intensidade de corrente na armadura, provocando assim movimento no motor (2).

A velocidade destes atuadores está dependente da diferença de potencial aplicada, enquanto que a corrente que o motor consome depende do binário requerido do mesmo. Nestes motores, devido ao contacto deslizante a alta velocidade das escovas de carvão com o comutador, é necessário assegurar a sua limpeza, prevenindo a acumulação de pó de grafite que pode causar curto-circuitos. Pela mesma razão, estes motores não podem ser utilizados em ambientes em que existe o perigo de explosão, uma vez que na sua utilização podem ser geradas faíscas durante a comutação (2).

Motores de passo são umas das opções mais atrativas para a atuação em ensaios mecânicos, uma vez que podem ser controlados diretamente por um computador ou

[1]Corrente Contínua
[2]Corrente Alternada

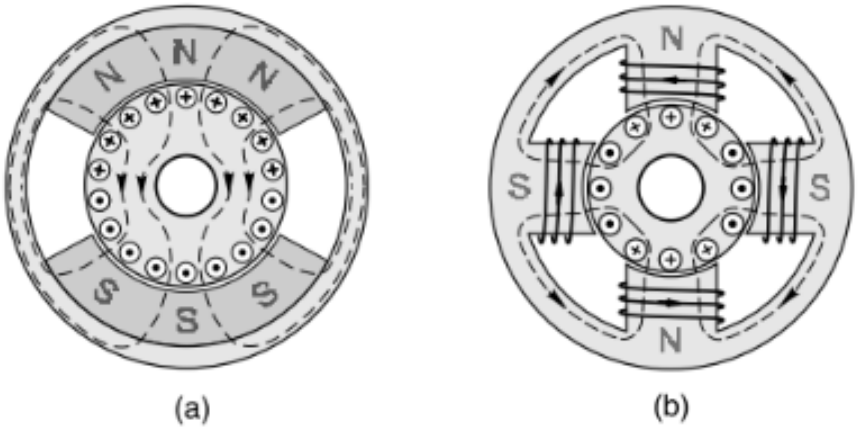

Figura 2.4: Motores de corrente continua: a) de 2 pólos e ímanes permanentes; b) de 4 pólos e bobines (2)

microcontrolador, e apresentam grande controlo de posição. O rotor destes motores é desenhado de modo a criar polos que se alinham com o campo magnético produzido pelos enrolamentos do estator, como é o exemplo do motor de passo de relutância variável visível na Figura 2.5 (2).

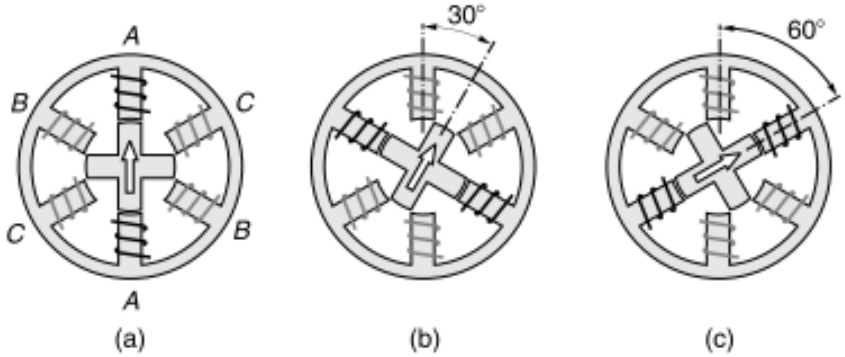

Figura 2.5: Princípio de operação de um motor de passo de relutância variável com um passo de 30° (2)

O rotor de motores de passo pode ser composto por material ferromagnético ou ímanes permanentes, enquanto o que o estator é sempre composto por bobines que através da sua ativação alteram o campo magnético no motor, provocando um alinhamento do rotor e assim o seu movimento (2, 20).

O modo de atuação do motor linear é o mesmo dos servomotores, mas elimina a necessidade de mecanismos intermediários de conversão de movimento. Estes motores possibilitam grandes acelerações e velocidades e exatidão de movimentos, embora necessitem de controlo em malha-fechada e de sensores de posição mais caros que os necessários no uso de um servomotor. Este tipo de motores, quando montados verticalmente, necessita de um sistema de contrabalanço para prevenir a queda da carga quando deixa de

ser alimentado, sendo esse sistema normalmente constituído por pesos e roldanas, molas ou amortecimento a ar, exercendo uma força oposta à força gravítica (20).

Dispositivos elétricos não são, contudo, a única opção de atuação de força. Fluídos, sendo gases ou líquidos, também podem utilizados para fornecer energia, deslocando objetos ou aplicando força (19). A automação industrial por meio de fluídos sob pressão dividiu-se em dois grupos bem definidos. Um primeiro grupo e certamente o de mais antiga aplicação pelo homem é o fluído hidráulico (fluído líquido sob pressão), e o segundo é o fluído pneumático (fluído gasoso sob pressão) (21).

Um atuador linear cujo funcionamento assenta numa solução hidráulica consiste num êmbolo conectado ao eixo de saída, introduzido num cilindro estanque (Figura 2.6). Quando o fluído é bombeado para a câmara A, o êmbolo e o eixo vão-se mover na direção de B, se o fluído for bombeado para a câmara B o êmbolo e o eixo vão-se mover na direção de A (19). Este princípio de atuação estende-se aos sistemas pneumáticos.

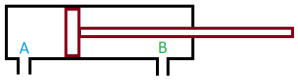

Figura 2.6: Cilindro hidráulico

Nos sistemas hidráulicos é necessária a existência de um líquido (geralmente óleo) para a sua operação, sendo este caro e em maioria dos casos perigoso no seu manuseamento e armazenamento, pelo que as tubagens têm de atuar como um sistema fechado, com o fluído a ser recolhido do reservatório para um lado do êmbolo, e a ser retomado ao reservatório pelo o outro lado do êmbolo. O fluído é puxado do reservatório por uma bomba, que produz a pressão necessária para a atuação do sistema, sendo necessário fazer a regulação da mesma para assegurar uma pressão constante ao longo do funcionamento e a correta filtragem para assegurar a integridade dos componentes a jusante. O movimento do cilindro é controlado por válvulas que direcionam o fluído hidráulico para a câmara requerida para estender ou recolher o cilindro (19).

Algumas das vantagens de um sistema hidráulico são o seu controlo superior de posição, velocidade e força, o facto de serem sistemas auto lubrificados e de terem uma relação entre peso, tamanho e potência consumida menor que os outros sistemas (22). Os atuadores alimentados por este tipo de sistema permitem forças bastantes altas e como não precisam de proteção contra sobre extensão do eixo permitem um deslocamento maior para o seu tamanho (19). A instalação deste tipo de atuadores apresenta, contudo, as desvantagens de elevado custo inicial para instalações sem sistema hidráulico prévio ou sem capacidade suficiente, apresentar perigos ambientais e maiores custos energéticos, devido à necessidade de transformação de energia elétrica ou química em mecânica e mecânica em hidráulica na bomba, e hidráulica em mecânica no atuador (22).

Apesar de ter os mesmos princípios de atuação, um sistema pneumático apresenta problemas e vantagens diferentes, provenientes das propriedades do ar e da sua obtenção e libertação.

Os sistemas pneumáticos, em geral, possuem uma pressão de operação inferior a sistemas hidráulicos, obrigando assim a atuadores com maior área de secção do pistão

para exercer a mesma força. A válvula que entrega ar ao cilindro opera de maneira similar ao seu equivalente hidráulico. Uma diferença notável é que o ar é gratuito, sendo que não é necessário devolver ao reservatório (19). Esta libertação, apesar de simplificar o sistema, significa constante descarga de ar pressurizado para o ambiente, gerando assim níveis de ruído elevados, ou o investimento em equipamento silenciador (21). O ar é extraído da atmosfera, sendo necessária filtragem para retirar parte das impurezas que o compõem e a humidade presente. A temperatura do ar aumenta ao passar no compressor, e como este é composto também por vapor de água, é necessário o seu arrefecimento para se dar a condensação. Apesar da compressibilidade do ar tornar necessária a existência de um reservatório para tornar a pressão mais constante, o seu controlo é relativamente mais simples que num sistema hidráulico (19).

Os atuadores pneumáticos apresentam vantagem em relação aos atuadores hidráulicos em relação ao preço, na simplicidade e compacidade da tubagem e nos efeitos ambientais em caso de falha (21, 23). No entanto, para além dos inconvenientes do ruído, necessidade de tratamento do fluído e compressibilidade do mesmo, esta solução também apresenta desvantagens no custo da implementação do sistema em instalações que não possuam já uma solução de ar comprimido instalada, a acrescentar à dificuldade de manter uma força constante e uniforme em atuadores pneumáticos e de ser necessário um atuador maior para aplicar a mesma força que um atuador hidráulico por operar com níveis de pressão significativamente inferiores (22).

Outra forma de atuação de força será um atuador piezoelétrico. O material cerâmico piezoelétrico usado num atuador piezoelétrico gera energia elétrica quando é submetido a energia mecânica (efeito piezoelétrico) e gera energia mecânica quando sujeito a energia elétrica. Um atuador piezoelétrico de múltiplas camadas (como se pode observar na Figura 2.7) é formado por filmes do material cerâmico de 100μm de espessura e filmes de elétrodo, o que lhe permite obter grande precisão de deslocamentos (3).

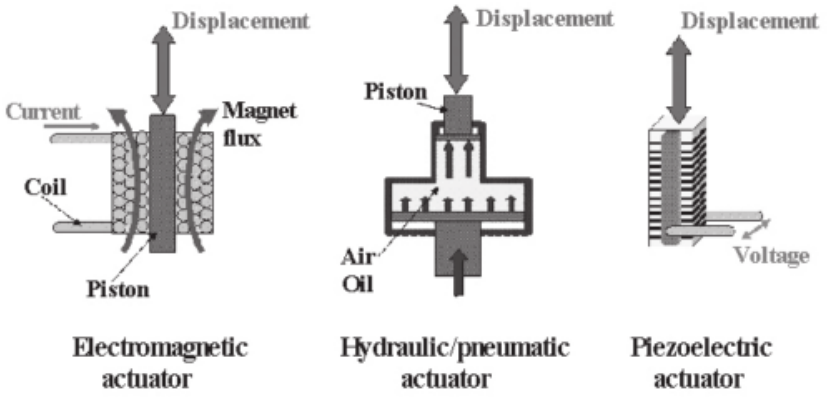

Figura 2.7: Atuador Eletromagnético, atuador hidráulico/pneumático e atuador piezoelétrico, respetivamente (3).

A aplicação destes atuadores é semelhante à dos atuadores eletromagnéticos, mas apresenta desvantagem no deslocamento que consegue exercer. Contudo estes atuadores apresentam uma maior eficiência de energia, assim como controlo da posição e força

(3). Apesar das suas vantagens este tipo de atuador é mais aplicado em situações de acionamento de curso reduzido e controlado, tal como acontece em aplicações de bio e nanotecnologia.

2.2.2 Medição de carga

Os sensores de força podem ser divididos em duas classes: contínuos e discretos. Um sensor quantitativo mede a força e representa esse valor através de um sinal elétrico analógico, como os extensómetros aplicados em células de carga. Os sensores qualitativos verificam limites de força ultrapassados e não o seu valor, apresentando menor fidelidade na representação do valor de força (8).

Para efetuar a medição da carga efetiva exercida é necessário um transdutor de força, ou seja, um sensor que reaja à mudança de carga, através de pressão ou deformação por exemplo, com um comportamento expectável e conhecido e que pode ser relacionado com o valor numérico.

Um extensómetro geralmente é um metal cuja resistência varia quando é deformado, permitindo assim, através da medição da alteração da sua resistência, obter a deformação aplicada no material em que está aplicado (8, 4). A relação entre a deformação e a mudança de resistência elétrica do extensómetro é expresso pelo fator de medida do extensómetro ("strain gauge factor"), apresentado na Equação 2.6 , onde R_0 é a resistência medida no elemento sem estar a ser exercida nenhuma deformação. O valor da resistência no extensómetro pode variar também com a temperatura, devido a efeitos de expansão ou compressão térmica do material que o compõe.

$$G = \frac{\Delta R}{R_0} \times \frac{1}{\varepsilon} \tag{2.6}$$

No entanto, as variações de resistência destes sensores de deformação não são fáceis de medir, uma vez que apresentam valores muito pequenos, pelo que se recorre ao uso de uma ponte de wheatstone (Figura 2.8) para obter um resultado passível de ser lido.

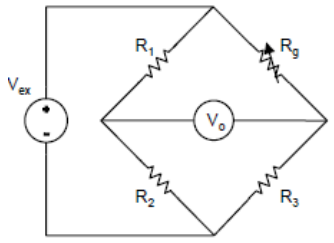

Figura 2.8: Ponte de Wheatstone (4)

Este circuito permite o cálculo de uma resistência variável (na Figura 2.8 a resistência R_g), tendo conhecimento das outras resistências que o compõem e da tensão (V_{ex}) aplicada, recorrendo à medição da tensão de saída (V_0) e à Equação 2.7 (4).

$$V_0 = [\frac{R_3}{R_3 + R_g} - \frac{R_2}{R_1 + R_2}]V_{ex} \tag{2.7}$$

Outro método de obter uma medição de força com aquisição de sinal elétrico é através do efeito piezoelétrico de alguns materiais, em vez do efeito piezoresistivo. Através deste efeito é possível obter uma variação de tensão elétrica através da deformação do material, ou seja, consegue converter uma mudança de força numa diferença de potencial elétrico (8). Este tipo de sensor apresenta o inconveniente de não fornecer sinal quando a força se mantém constante, sendo o seu uso mais útil para forças com variação rápida (4).

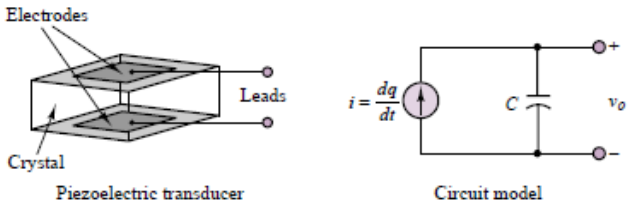

Figura 2.9: Transdutor piezoelétrico (5)

Este tipo de sensores consiste num cristal piezoelétrico e dois elétrodos. Embora, em princípio, seja possível utilizar um amplificador de diferença de potencial para amplificar o sinal obtido do transdutor, é muitas vezes mais vantajoso utilizar amplificador de carga (Figura 2.10), que é basicamente um circuito integrador caracterizado por uma alta impedância (5).

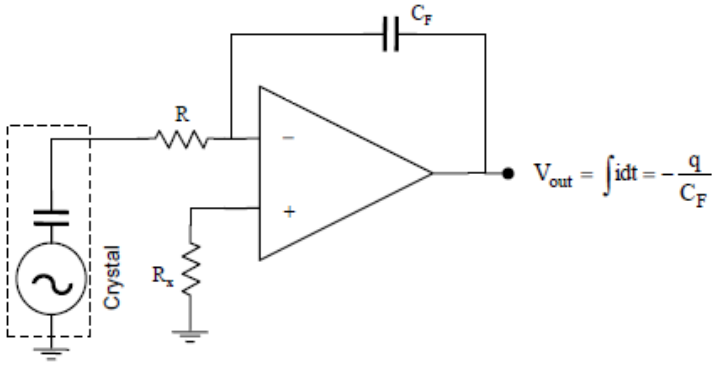

Figura 2.10: Amplificador de carga (4)

Outros sensores de força podem atuar tendo em conta o efeito capacitivo. Um transdutor de pressão capacitivo, como o mostrado na Figura 2.11, baseia o seu funcionamento na deformação de uma membrana de silicone, que serve como um dielétrico do condensador, sendo os elétrodos peliculas de metal distanciadas alguns microns (24).

A deformação da membrana, dependente da força aplicada, aproxima os elétrodos e altera a capacidade elétrica do sistema de acordo com a Equação 2.8. Contudo, sensores capacitivos são inerentemente não-lineares e a medição de pequenas capacidades de uma estrutura miniaturizada é bastante difícil, devido a efeitos parasitas e interferências eletromagnéticas do ambiente (24).

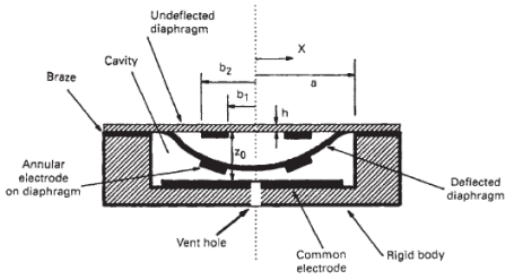

Figura 2.11: Transdutor capacitivo de pressão (6)

$$\text{capacidade } (C) = \frac{\text{Constante Dielétrica } (D) \times \text{Área}(A)}{\text{Distância entre elétrodos } (d)} \tag{2.8}$$

2.2.3 Medição de posição

A medição de movimento e posição é talvez um dos problemas mais usualmente encontrados em medição em engenharia. As medições de interesse incluem a posição relativa e absoluta, velocidade e aceleração. Apesar dos métodos de medição poderem ser de translação ou de rotação, os princípios que os regem são semelhantes, baseando-se tanto em mudanças de propriedades elementares, como a resistência do material, em campos elétricos ou magnéticos, bem como em sistemas de materiais com características especiais e sistemas óticos ou de visão (5), havendo por isso bastantes soluções disponíveis no mercado, pelo que se vão explorar apenas algumas das mais interessantes para a problemática de medição de extensão em ensaios materiais.

A medição de posição e deslocamento de objetos é essencial para diversas aplicações, como processos de controlo em malha fechada, avaliação de desempenho da atuação, controlo de tráfego de transporte, robótica, entre outros. Por leitura de posição entende-se como determinação da coordenadas de um objeto, sejam elas lineares ou angulares, em relação a uma referência (8).

Um encoder pode ser definido como um dispositivo que converte movimento mecânico em sinais elétricos usados para monitorizar a posição ou velocidade. Um encoder permite o seu uso quer em sistemas lineares quer rotativos (5, 7).

Um encoder de posição consiste num disco (para movimentos rotativos) ou régua (para movimentos lineares), que contém áreas pretas e brancas, ou opacas e translúcidas, que estão dispostas para reproduzir um código indicador de incremento unitário de deslocamento ou de posição, como o representado na Figura 2.12. Um conjunto fixo de foto-díodos serve de sensor de feixe luminoso, que devolve uma diferença de potencial correspondente ao valor binário das posições na fita ou disco (5).

Existem dois tipos básico de encoders: os encoders absolutos (Figura 2.13) devolvem um código binário completo por cada posição na sua saída digital, e são geralmente utilizados em aplicações onde é necessário saber a posição atual. Os encoders incrementais

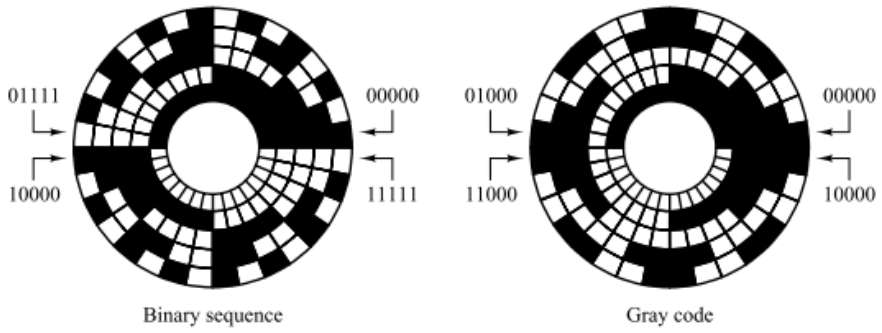

Figura 2.12: Código binário e Gray, respetivamente da esquerda para a direita, para encoders absolutos, angulares de posição (5)

(Figura 2.14) produzem um pulso por cada incremento de posição, e a posição é calculada tendo em conta a resolução do encoder, e o somatório dos seus incrementos desde um ponto de início conhecido (8).

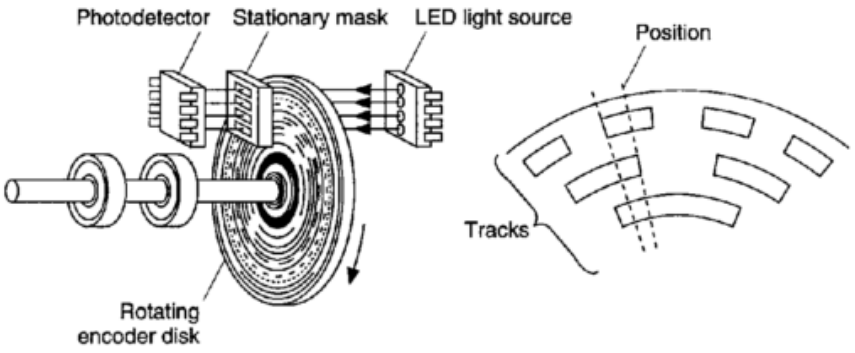

Figura 2.13: Representação do funcionamento de um encoder ótico absoluto (7)

Outro dispositivo usualmente utilizado para obter posição angular é o resolver. O princípio de funcionamento destes dispositivos assenta na transformação de energia mecânica rotativa em energia elétrica, assemelhando-se a um motor elétrico funcionando como gerador (18). Um resolver tem dois estatores desfasados 90° entre si, o que provoca nos sinais de saída uma diferença de potencial induzida, proporcional ao cosseno e ao seno do ângulo do rotor (5, 7).

Um transdutor de posição ou deslocamento pode ser construído também com o fundamento de atuação de um potenciómetro linear ou rotativo, cujo princípio de funcionamento se baseia no facto de a resistência aumentar com o comprimento do condutor (8, 7).

Neste tipo de sensor, a variação de posição faz variar o comprimento do filamento resistivo no circuito, que por sua vez altera a resistência entre os terminais. Em vários dispositivos, a leitura da resistência é substituída pela leitura da variação de diferença

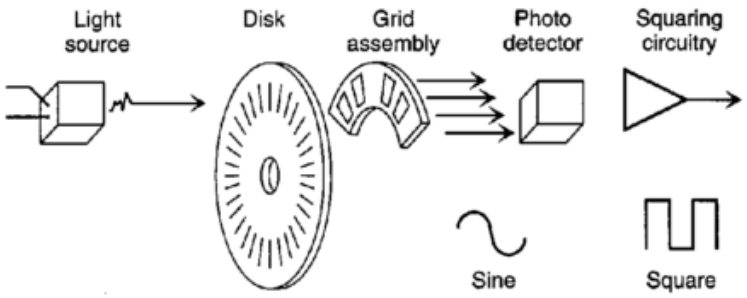

Figura 2.14: Representação do funcionamento de um encoder ótico incremental (7)

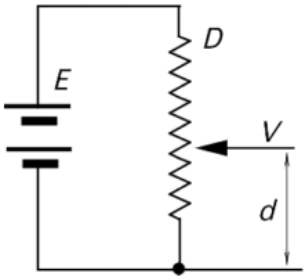

Figura 2.15: Potenciómetro como sensor de posição (8)

de potencial, que por sua vez é proporcional ao deslocamento segundo a Equação 2.9 (8).

$$V_{\text{saída}} = E_{entrada} \times \frac{\text{deslocamento}}{\text{Deslocamento Máximo}} \tag{2.9}$$

Apesar de serem bastante úteis em algumas aplicações, este transdutor de posição tem algumas desvantagens, como a fricção entre componentes, aquecimento do potenciómetro devido à desvantagem anterior e à diferença de potencial aplicada no mesmo, a necessidade de estar conectado fisicamente ao componente cuja deslocação se quer medir, e a sua baixa velocidade.

Existem ainda soluções que não envolvem contacto direto com o objeto a ser medido, como é o caso da visão assistida por computador e vibrometria laser. Apesar da primeira solução necessitar de mais poder computacional, oferece vantagens, como a possibilidade de observação do teste à posteriori, proporcionando a oportunidade de uma observação mais atenta e detalhada da reação do provete de material às tensões aplicadas. Através de novos métodos desenvolvidos na área, é também possível com este recurso analisar áreas específicas do material e compreender a resposta local à carga aplicada no provete, e estudar assim o efeito de diferentes geometrias ou defeitos no comportamento do material em semelhantes situações. Esta solução traz consigo também as desvantagens referentes a qualquer problema de visão por máquina, como o custo do equipamento e os cuidados

necessário para a obtenção de imagem, como a luminosidade e reflexos.

Capítulo 3

Análise da solução prévia

O primeiro passo na compreensão das possíveis funcionalidades da máquina de ensaios será o estudo da mesma, analisando o *hardware* utilizado no controlo e obtenção de dados e o respetivo *software*, com especial atenção a modos de funcionamento, processamento de sinais e protocolos de comunicação. Esta análise irá permitir avaliar vantagens e desvantagens do sistema atual, fomentando propostas de melhoria e estruturando prioridades, a serem descritas posteriormente em comparação com o *software* e *hardware* desenvolvidos neste projeto.

3.1 Funcionamento do sistema: Análise geral

O funcionamento da máquina de ensaios consiste no controlo de um atuador linear em deslocamento ou força, que pretende causar deformações num provete de teste, e que devolve para cada instante os valores da força aplicada, deslocamento e frequência. O funcionamento pode ser analisado esquematicamente na Figura 3.1 e será descrito mais à frente neste capítulo.

Apesar das propriedades do material não poderem ser medidas na sua maioria de modo direto, podem ser extrapoladas através de cálculos, sendo necessários os valores da força e do deslocamento para realizar ensaios de tração e compressão ao material. No caso da máquina de ensaios em análise estes valores podem ser ambos obtidos do atuador, sendo o deslocamento obtido diretamente através de um encoder interno e a força obtida indiretamente através da medição da corrente nos enrolamentos do motor (25).

Os valores de deslocamento são recebidos pela placa de *Arduino* através de uma derivação no cabo proveniente do encoder interno do atuador, identificado na Figura 3.2. O módulo *Arduino* possui uma placa de leitura de célula de carga para descodificar o sinal proveniente da mesma, e assim obter também a força exercida nesse instante. Estes valores são enviados de seguida por RS-232 para o computador através de porta série, e são lidos com recurso a uma porta virtual para posterior análise, sendo a mesma processada com recurso a outros softwares, como MATLAB ou Excel.

Após ter sido realizada uma análise superficial e resumida e obtido a perspetiva geral

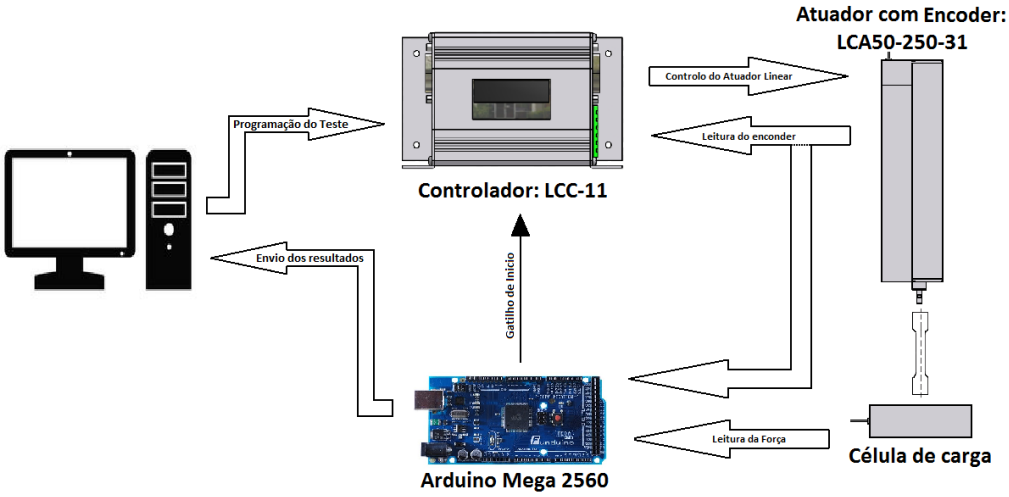

Figura 3.1: Diagrama de funcionamento geral da máquina

Figura 3.2: Montagem dos componentes, pormenor da ligação encoder-LCC e encoder-*Arduíno*

do sistema e a sua contextualização no âmbito de investigação é necessário realizar um aprofundamento do estudo, recorrendo para isso a uma separação teórica de módulos para facilitar a compreensão da máquina. A análise desta irá focar-se nos módulos individuais que a compõem: módulo *Arduino*, célula de carga, controlador, atuador e computador, visíveis na Figura 3.3.

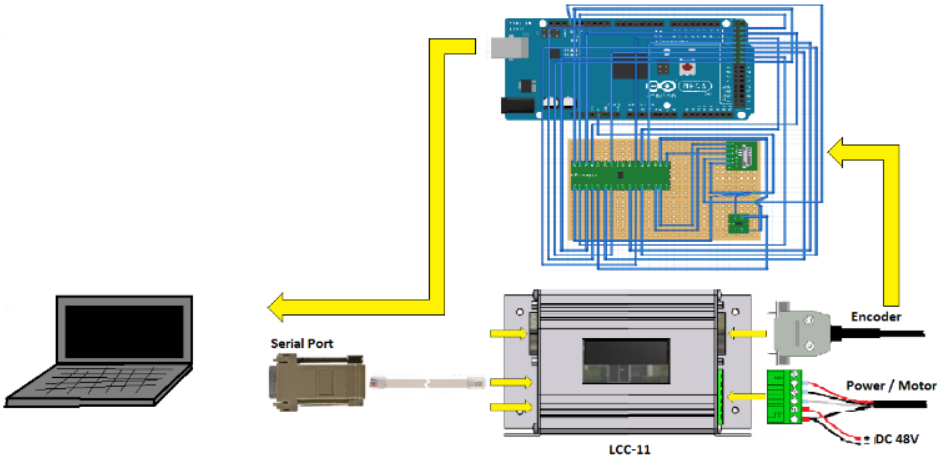

Figura 3.3: Esquema geral de ligações

3.2 Obtenção de dados de teste: Módulo baseado em solução Arduino

O módulo *Arduino* tem como base uma placa *Arduino Mega 2560*, que é constituída pelo microcontrolador ATmega2560 alimentado a 5V e com um oscilador interno de 16 MHz (26), e que contém 54 pinos digitais que podem ser usados como entradas ou saídas e que permitem uma tensão de entrada até 12 V, 16 dos quais podendo ser utilizados como entradas analógicas (27), utilizando um cabo USB de ligação com o computador tanto para comunicação como para alimentação de potência, identificado na Figura 3.4. O microcontrolador é utilizado para a recolha de informação de envio da mesma por porta série para o computador durante o ensaio, nomeadamente os valores de posição (POS), força (AN), e frequência (Freq), sendo este último valor calculado pelo próprio através do intervalo de tempo entre envios. O envio da informação é acionado pela receção de corrente na sua entrada digital A21, proveniente da saída digital 0 do controlador, fisicamente identificada como o pino 9 da ficha DB-26 do mesmo (28).

Para a obtenção do valor de posição o microcontrolador possui uma *shield* de processamento de sinal proveniente do encoder. Esta *shield* permite a montagem de um oscilador e de uma interface intermédia entre o encoder incremental e o microprocessador, nomeadamente o HCTL-2032 (29), conectando assim os sinais do encoder aos pinos A22 a A37 do microcontrolador. Para processamento dos mesmos o microcontrolador

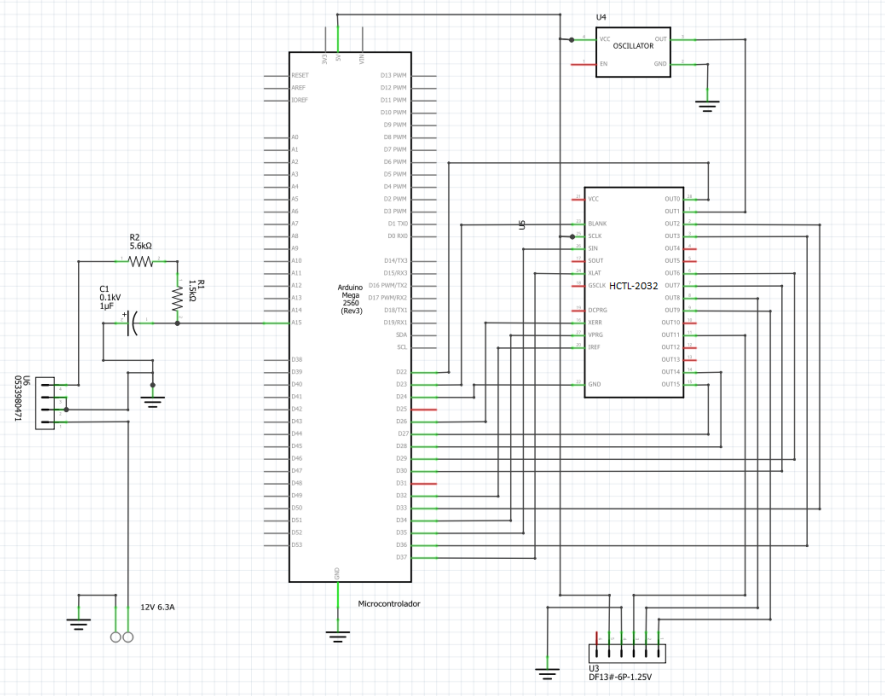

Figura 3.4: Módulo *Arduino* e respetivas ligações

está dotado de uma biblioteca específica do descodificador de encoder, que possui funções que permitem retirar o valor da posição diretamente. Por uma questão de simplificação, e não tendo sido detetada necessidade para mais, o HCTL só recebe quatro ligações provenientes do encoder, nomeadamente as ligações Z+, B+, A+, e terra. As ligações provenientes do encoder apresentam-se fisicamente identificadas nos pinos 2, 3, 1, 12 e 13 respetivamente da ficha DB-15 do mesmo, estando os últimos dois pinos identificados como terra (28). O valor enviado pelo módulo *Arduino* permite extrair a posição ou deslocamento sabendo que a resolução o encoder é de 1μm por incremento; Para a obtenção do valor de força é utilizada uma segunda *shield* montada no *Arduino Mega* e um circuito externo, necessário para transformar os valores recebidos da célula de carga em sinais que possam ser tratados pelo circuito e pelo microcontrolador. Tratando-se a célula de carga nada mais que uma ponte de *wheatstone* (Figura 3.5) em que a resistência variável é provocada pela deformação de extensómetros, o circuito externo é necessário para alimentar a célula e modificar e amplificar as tensões recebidas das saídas da ponte (Na imagem Output+ e Output-) num só sinal analógico, compreendido entre 0.5 e 4.5V.

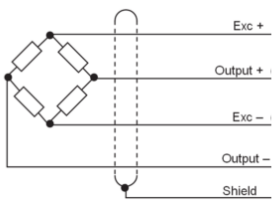

Figura 3.5: Esquema de uma célula de carga. Retirado de (9)

A alimentação é feita na *shield* através dos contactos identificados na Figura 3.3, enquanto que a ligação com o circuito externo é realizada com o uso de uma ficha, sendo o sinal de saída que, depois de tratado, vai ligar à entrada analógica A15 do *Arduino* o pino da esquerda, sendo os restantes destinados à alimentação. A *shield* possui ainda um circuito RC[1] que atua como filtro passa-baixo, atenuando o efeito de sinais de frequências mais elevadas que possam surgir como ruído e cortando sinais com uma frequência superior a 22 Hz.

O valor a enviar de leitura da célula de carga é uma média de 5 valores lidos na entrada analógica, ao qual se subtrai um valor anteriormente configurado de calibração (Equação 3.1).

$$\text{Digital Average} = \frac{\sum_{i=1}^{5} \text{Digital Input}}{5} - \text{Digital Offset} \tag{3.1}$$

O valor de tensão recebida na entrada analógica do *Arduino*, não podendo ser lido diretamente, é calculado dividindo o valor de alimentação da célula de carga pelo número máximo que a leitura pode atingir (10 bits=2^{10}=1024), multiplicando o resultado da divisão pelo valor médio lido na entrada analógica (Equação 3.2):

$$\text{Analogic Input}[V] = \frac{5V}{1024} \times \text{Digital Average} \tag{3.2}$$

O valor da força exercida da célula de carga é calculado recorrendo à calibração da mesma, criando um gráfico da força em função do valor de tensão elétrica calculado pelo microcontrolador, como o apresentado na Figura 3.6, extraindo assim a Equação 3.3. Esta calibração tem de ser feita sempre que se altera a célula de carga, podendo ser necessária a troca por razões de limites de cargas admitidas, exatidão de leitura, tipo de ensaio e de leituras, sendo necessário realizar a alteração da equação resultante no programa *Arduino*, obrigando o utilizador a ter conhecimentos de programação neste ambiente.

$$\text{Carga}[N] = 24,797 \times \text{Analogic Input} - 11,04 = 24,797 \times \frac{5V}{1024} \times \text{Digital Average} - 11,04 \tag{3.3}$$

3.3 Atuação e controlo

O controlador em análise (LCC-11[2] , que para efeitos de análise de modos de funcionamento, e de resto bastante semelhantes, se considera igual ao controlador LCC-10[3]) da SMAC tem possibilidade de atuar de seis modos (11)

[1] Circuito composto por uma resistência e um condensador
[2] Single axis brushless controller, built-in amplifier, 16-bit analog output
[3] *Single axis brushless controller, built-in amplifier, 10-bit analog output*

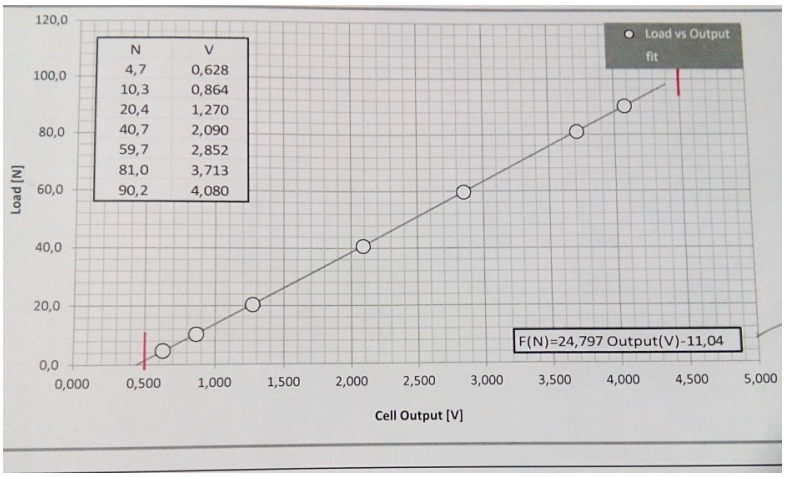

Figura 3.6: Gráfico de calibração da célula de carga

- Modo de controlo em malha aberta: Este modo permite a excitação do motor sem recurso a sensores para devolver a posição final, controlando apenas a tensão aplicada, não sendo o modo mais interessante de controlo para os ensaios a realizar.

- Modo de localização de posição inicial: Este modo permite vários métodos para localizar a posição inicial pré-definida ou a referência absoluta do sistema, sendo interessante e necessário na inicialização dos ensaios, sendo que o atuador usado (LCA50-250-31-FSE3) possui apenas um encoder incremental, ou seja, sabe apenas o deslocamento, sendo sempre necessário o conhecimento deste ponto para referência da posição.

- Modo de perfil de posições: Este modo permite controlar a posição do sistema, podendo limitar ao mesmo tempo a velocidade e aceleração imposta nos deslocamentos.

- Modo de perfil de velocidades: Este modo é utilizado para controlo da velocidade do sistema sem consideração pela posição (tendo em atenção a limitação dos extremos físicos da máquina). Uma vez que o atuador não consegue obter de modo direto as velocidades impostas, faz o controlo das mesmas obtendo a posição e extrapolando a informação, permitindo um controlo em malha fechada com o auxilio de PID[4]. Um controlador PID permite ao sistema atenuar o erro de atuação em estado estável a partir de ação integral e antecipar distúrbios no sinal através de ação derivativa, mas por agir em função de um erro medido pode não ter resposta rápida o suficiente para o tipo de acionamento requerido. Para corrigir esta possível fragilidade o controlador possui um compensador *feedforward* de posição e velocidade, que, e uma vez que os parâmetros de exame já estão definidos à *priori*, vai atuar sobre um erro expectável, isto é, vai atuar sobre perturbações antes de as mesmas induzirem erro no sistema (30). Este modo de atuação é comum com os modos de perfil de posição e de força, podendo ser visualizado na Figura 3.7.

[4]Controlador Proporcional, Integral e Derivativo

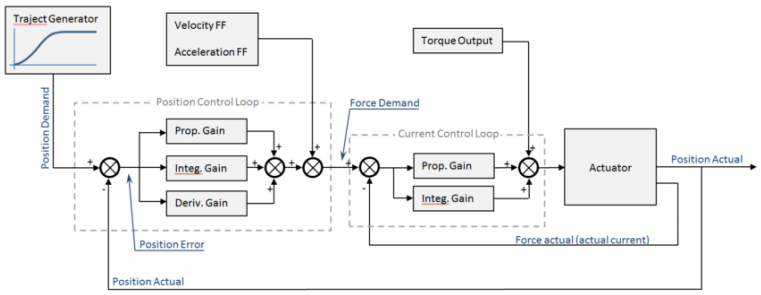

Figura 3.7: Esquema do controlo de posição feito pelo controlador (10)

- Modo de perfil de forças: Este modo permite o controlo do ensaio através da monitorização da força exercida pelo sistema (Figura 3.8). Tal como os dois modos anteriores possui funções que delimitam o final de ensaio. O controlador consegue calcular a força desenvolvida pelo atuador através da obtenção da intensidade de corrente nos terminais do motor, comparando-a numericamente com a corrente teoricamente necessária sem escorregamento nos seus enrolamentos e controlando os índices de fricção na vara do atuador e de atuação gravítica na massa com deslocamento, e assim fazer um controlo de atuação em malha fechada, permitindo atuar com mais precisão (25).

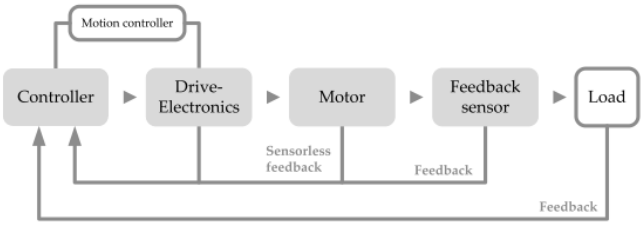

Figura 3.8: Diagrama de controlo (11)

- Modo de posição cíclica: Este modo permite o controlo sem recorrer a perfis de posição internos, podendo limitar a mesma.

O atuador utilizado, como acima referido, é do tipo elétrico linear, possuindo um curso de 250mm e uma força máxima de 90N quando alimentado a 48V, tendo também a possibilidade de ser alimentado a 24V. Em atuação constante consegue manter uma força máxima de 36N. Este equipamento traz já instalado um encoder incremental com uma resolução de 1μm e está instalado na posição vertical, movendo-se no sentido positivo com a deslocação descendente e no sentido negativo com deslocações ascendentes.

3.4 Interface de programação de testes: Módulo baseado em software de controlo SMAC

No inicio de um novo teste, o utilizador terá de abrir o ambiente integrado de comunicação com o controlador: "*SMAC Control Centre*"; onde estão disponíveis as opções principais de modificar parâmetros de teste e envio do código de comandos para o controlador LCC, de execução de programas criados e de ajustes do sistema. Na funcionalidade de edição de funções, aqui denominadas por "macros" (Figura 3.9, visível na área delimitada e legendada por 3), é possível a modificação ou criação das mesmas, sendo possível incutir-lhes funções de chamada de outras macro ou repetição de si mesma, funções que controlem o movimento através dos modos de atuação anteriormente referidos, funções de condicionamento de atuação (funções "*IF*"), e funções de leitura e escrita de variáveis. O controlador permite a criação de macros com numeração de 0 a 59, estando 4 macros (60 a 63) reservadas a funções do sistema (12). Apesar de este número parecer baixo, encontra-se recomendado na bibliografia que a realização de um programa não implique o chamamento de mais de 9 macros consecutivas (31).

Depois de criadas ou modificadas as macros necessárias ao funcionamento do atuador do modo pretendido para o ensaio em questão, é possível guardá-las no controlador, fazendo o download. Uma vez feito o download das macros para o controlador é possível executá-las individualmente, isto é, se o utilizador quiser correr um programa, e se este se encontrar bem feito, só necessita de correr uma macro, que esta irá chamar as que necessitar para a continuação do ensaio. Isto significa também que o utilizador pode correr mais do que tipo de ensaio sem ter de fazer a modificação ou o download de macros para o controlador, sendo apenas necessário se houver alteração de parâmetros. Na Figura 3.9 podemos ver a o ambiente de controlo, onde está identificada a zona de edição de funções/macros com o número 1, a zona de comunicação com o controlador com o número 2, onde é possível fazer o *download* de macros, criação e eliminação de programas, e por fim, a zona onde é possível visualizar, selecionar e comentar macros criadas, identificado com o número 3 (12).

Para iniciar a execução do programa gravado basta selecionar a macro de inicialização do mesmo e a porta de comunicação do computador com o controlador, sendo possível neste caso correr programas em quatro controladores simultaneamente, nas opções disponíveis o canto inferior direito da figura 3.10, na área delimitada e nomeada pela legenda "Macro". Depois de selecionado o controlador e a macro a correr basta selecionar o botão "*Run*" correspondente. Quando o programa iniciar a sua execução o controlador entrará em comunicação com o módulo *Arduino*, enviando um pulso a cada 100ms, que inicializa a sua tarefa de leitura de valores e respetiva comunicação ao computador. Este separador do ambiente de controlo do *hardware* da SMAC também permite a visualização em tempo real de até 4 variáveis, adicionadas na área delimitada pela legenda "*Logging*" na Figura 3.10, e o seu registo num ficheiro .csv (ficheiro de texto separado por vírgulas (32)) , sendo necessário para isso selecionar um intervalo de tempo entre leituras, o número de leituras a realizar e pelo menos uma variável que se pretende ler, sendo as principais a posição, velocidade e força atuais e o erro de seguimento, mas sendo possível selecionar variáveis de uma biblioteca com mais de 90 entradas.

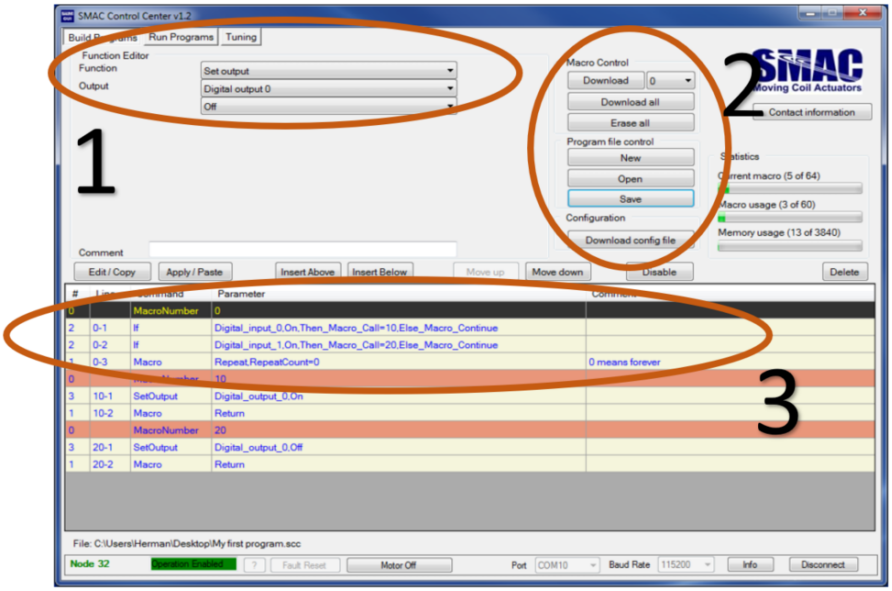

Figura 3.9: SMAC Control Center v1.2, opção de criação de programas (12) 1- Criação e edição de funções 2- Comunicação com o controlador 3- Exemplo de macro

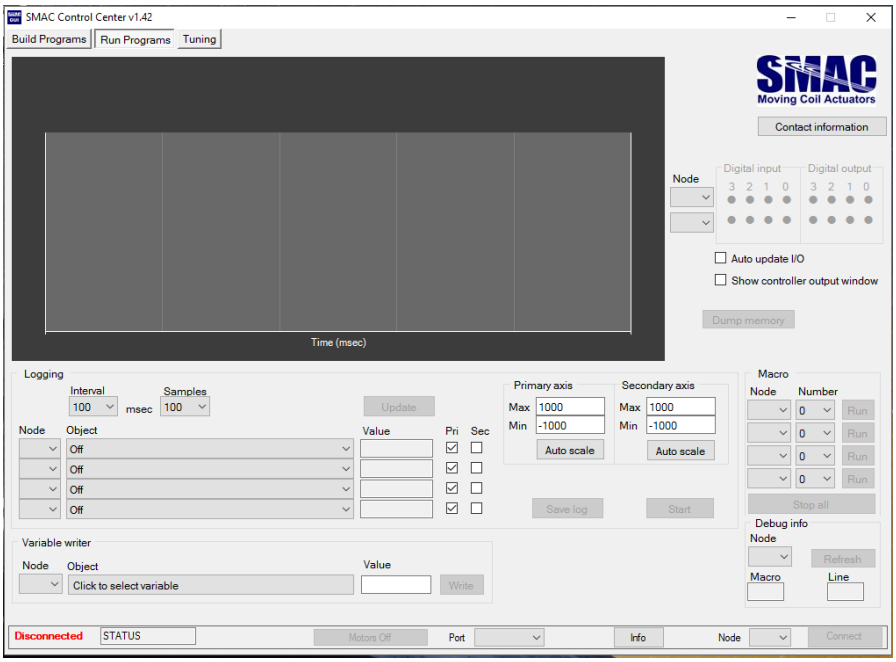

Figura 3.10: *SMAC Control Center v1.42*, opção de execução de programas

Para ajuste do sistema o ambiente de controlo, de modo a conseguir minorar o erro nos ensaios, tais como erro associado a mudança de peso na massa com deslocamento do atuador, uso de molas ou diferentes orientações de montagem, o software possui um separador que permite a modificação de parâmetros necessários na execução de programas, vistos na Figura 3.11, tais como valores para o controlador PID de posição e velocidade, para o controlador PI[5] de força e para os limites físicos do atuador, sendo ainda possível realizar um teste de confirmação (12). Os valores dos parâmetros apresentados na Figura 3.11 são apresentados quando o utilizador conecta corretamente o computador com o *software* ao controlador, sendo assim possível verificar e modificar de maneira bastante conveniente e facilitada os mesmos, implicando uma maior monitorização em ambiente académico onde o acesso pode não estar restrito.

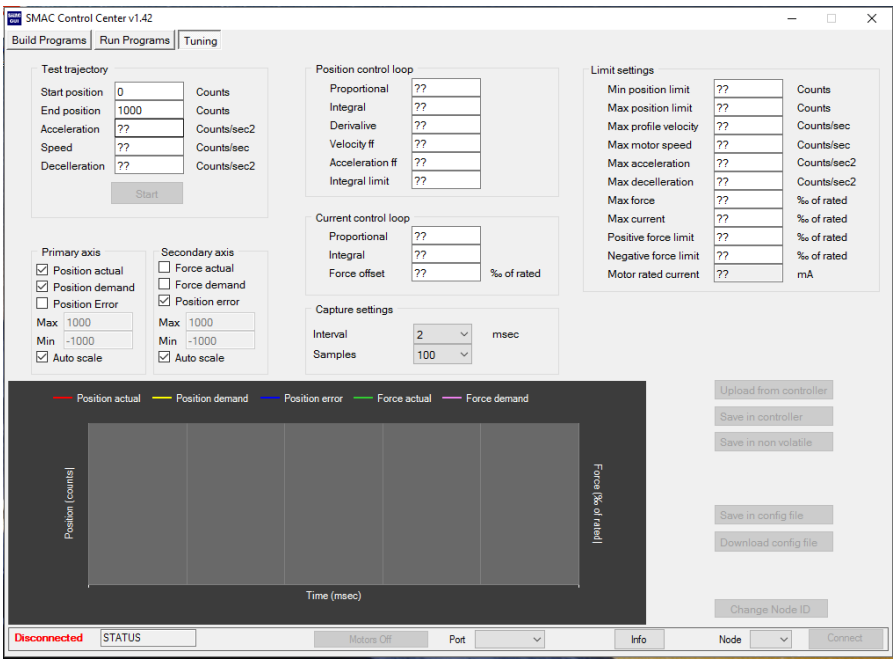

Figura 3.11: *SMAC Control Center v1.42*, opção alteração de parâmetros

3.5 Análise de um teste

3.5.1 Iniciação do controlador LCC e atuador

Para facilitar a compreensão do sistema e continuar a explicação do seu funcionamento, vai-se proceder à análise de um exemplo de ensaio cíclico, apresentando os passos para a sua execução e respetiva análise. O exemplo apresentado representa um ensaio real e código previamente realizado, pelo que para esta análise se vai ignorar os comentários definidos pelo seu criador e se vai apresentar a justificação dos passos do programa.

[5]Controlador Proporcional e Integral

No intuito de melhor compreender todos os dispositivos componentes no sistema, além da utilização de *software* de leitura de porta série definida para comunicação com o *Arduino*, recorreu-se também à leitura da porta série de comunicação com o controlador (Figura 3.12), permitindo assim não só relacionar o estudo de tipo de comunicação teórica presente nos manuais da marca (11, 33) com uma metodologia mais prática, mas também para facilitar a compreensão de estrutura de mensagem.

Baudrate	*115200 bps*
Data bits	*8 bits*
Parity	*None*
Stop bits	*1 bit*
Flux control	*None*

Figura 3.12: Configuração da comunicação Rs232 com o controlador

Após iniciação e correta conexão dos dispositivos integrantes no sistema, dá-se uma comunicação do controlador com o computador para assegurar sincronismo de configurações, indicando assim fatores importantes, como versão de *hardware* e *software*, nome de dispositivo, etc. (Figura 3.13).

Tendo em consideração o facto de o atuador LCA50-250-3 ser constituído por um encoder incremental, e tendo como seu elemento originário de força motriz um motor linear de múltiplos polos, torna-se necessário, após iniciar o sistema, descobrir a posição inicial e a fase em que se encontra, estando este procedimento implementado na macro 0, apresentada na Figura 3.14.

O comando de deteção de fase implica movimentos bruscos da haste do atuador, pelo que antes de se proceder à execução desta macro é necessário confirmar se não está nenhum provete montado na máquina, sendo o impulso gerado pelo atuador possivelmente danoso para o sistema e para o material em teste. Após detetadas as fases do atuador este irá procurar o seu ponto de referência de posição, a partir do qual, e juntamente com o conhecimento de número de incrementos do encoder interno irá habilitar o controlador a conhecer os seus deslocamentos. A haste do atuador vai-se mover de seguida para a sua posição inicial.

Esta macro tem desativadas duas linhas: A primeira se ativada permite o programa correr a função de "*Homing*" sem proceder à deteção de fases; A segunda linha desativada e última da macro em análise, se ativada permite saltar diretamente para a macro número 10, que inicia o ensaio. Esta última linha encontra-se desativada não só para permitir correr esta função para qualquer ensaio sem ser necessária alteração do programa, mas também para permitir o utilizador montar o provete de teste.

3.5.2 Iniciação da leitura de resultados

Após montagem do provete e de asseguradas as condições para inicio de teste é necessário iniciar a comunicação entre o *Arduino Mega* e o computador utilizado, recorrendo para isso ao *software Cool Term*, que lê o conteúdo recebido pela porta série configurada

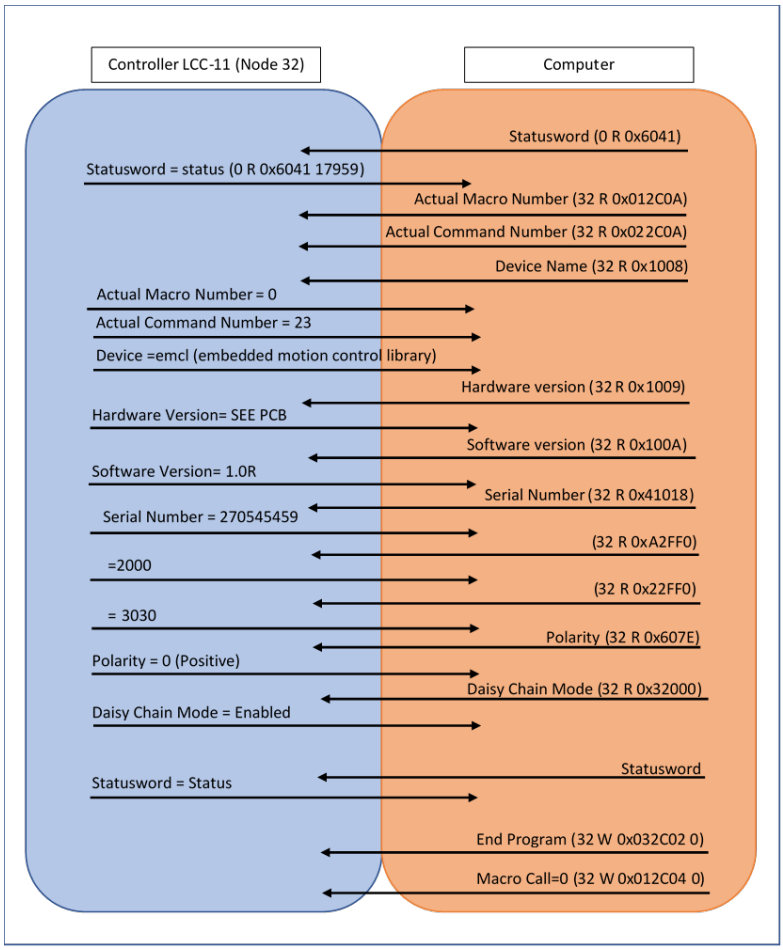

Figura 3.13: Diagrama de interações entre o computador e o controlador na iniciação da ligação

0		MacroNumber	0	
10	0-1	PhaseDetect	Forced,Time=,Current=,Tolerance=	Phase detect - forced
		PhaseDetect	Initial_position_allways_known,Rotorpos=0	Phase detect - forced
5	0-2	Homing	Endstop_and_indexpulse,Negative,Acc=,Vendstop=,Force=,Vindex=,Timeout=,Off...	
3	0-3	PositionMove	Absolute,Target=0,Vel=,Acc=,Change_immediate	Go to position 0
1	0-4	Wait	Time,Timeout=2000	
4	0-5	PositionMove	Absolute,Target=170000,Vel=50000,Acc=,Change_immediate	
		Macro	Jump,MacroNumber=10	

Figura 3.14: Macro 0, Função de *Homing*

no computador e grava num ficheiro de texto toda a comunicação, sendo a janela de trabalho deste software visível na Figura 3.15.

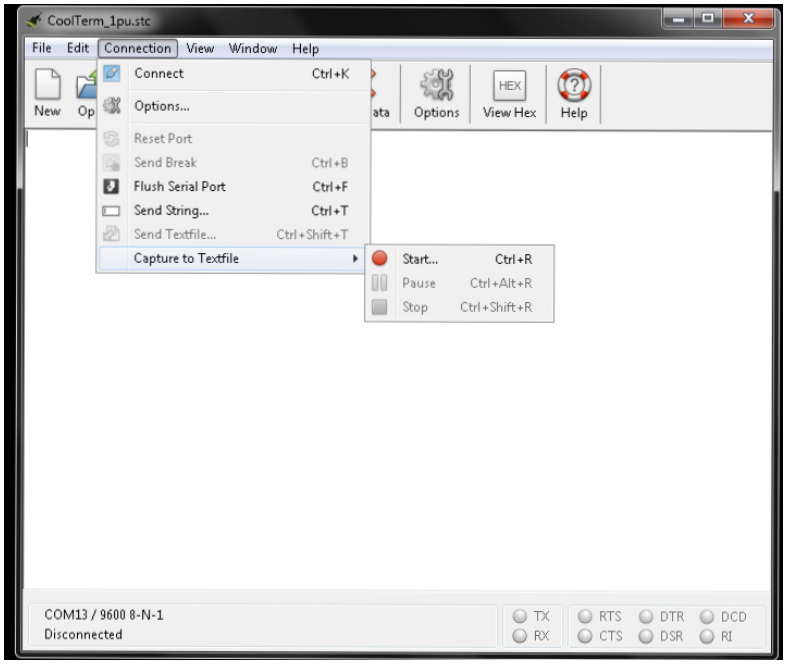

Figura 3.15: *Software* de comunicação com o *Arduino Mega* através da porta série

Depois de estabelecida a comunicação e estabelecidos os seus parâmetros, não só de comunicação, mas também de identificação do ficheiro gerado, o *Arduino* vai enviar uma mensagem a pedir a escrita do caracter "s", para iniciar a sua leitura do encoder e célula de carga, assim como cálculo da frequência. Estes valores são enviados por ele para o computador sempre que recebe do controlador um impulso da saída digital 0 do último, estando a comunicação *Arduino*-LCC assegurada por esse único sinal.

3.5.3 Programação de um teste de atuação cíclica

0		MacroNumber	10	fase 1
2	10-1	SetVariable	Var=General_purpouse_registers-W25(0x192C00),Variable,Var1=Position_actual...	Records actual max torque on W25
3	10-2	SetVariable	Var=General_purpouse_registers-W23(0x172C00),Subtract,Variable_and_consta...	Subtracts 20000 (=2cm) to W25 and stores result on W23
1	10-3	GetVariable	Var=General_purpouse_registers-W60(0x3C2C00)	Reports W23 value to the serial port
1	10-4	Wait	Time,Timeout=75	timeout=150
1	10-5	GetVariable	Var=General_purpouse_registers-W23(0x172C00)	Reports W23 value to the serial port
1	10-6	Macro	Jump,MacroNumber=11	

Figura 3.16: Macro número 10

A macro número 10, apresentada na Figura 3.16, está programada para ser o inicio deste ensaio, sendo apenas necessário para o iniciar correr esta função. Esta macro irá

realizar essencialmente o reconhecimento da localização inicial do provete e a irá obter variáveis necessárias para o funcionamento do ensaio:

1. 1. Grava o valor da posição inicial na posição de memória W25, acedido através da função "*Position actual value*" (valor de posição inicial) e apresentado na Figura 3.17;

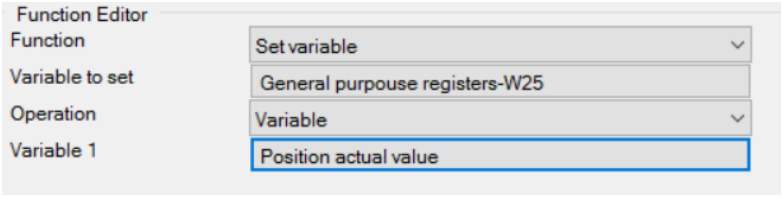

Figura 3.17: Função para guardar o valor inicial de posição da variável W25

2. Grava o resultado da subtração de 2cm ao valor da posição inicial na posição de memória W23 para definição de ponto final de ensaio através da função "*Subtract, Variable and constant*" (subtração, variável e constante), como apresentado na Figura 3.18;

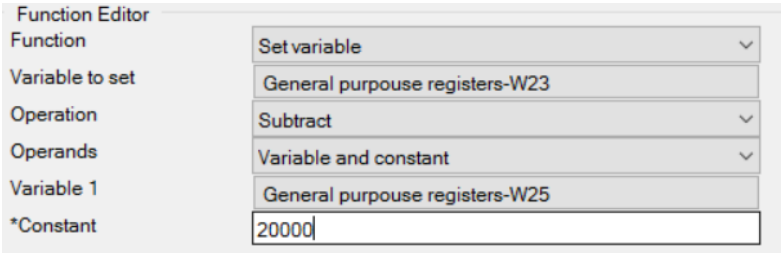

Figura 3.18: Função para guardar o resultado da subtração da posição inicial por a constante equivalente a 2cm

3. A função "*Get Variable*" permite ao controlador enviar por porta série o valor que está guardado na variável W60, que tem como conteúdo o caracter "H". A opção de alterar o valor destes registos sem ser por programação apresenta-se no separador "*Run Programs*" e pode ser visto na Figura 3.19. O envio de caracteres por porta série encontra-se no programa como resquício de programação num controlador semelhante da mesma marca (controlador LAC-1), que o presente controlador (LCC-11) veio substituir;

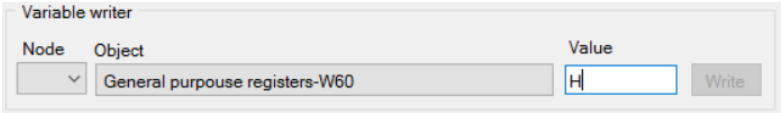

Figura 3.19: Função de escrita de valor no registo W60

4. Atrasa o decorrer do programa para a obtenção do valor requerido. Os tempos definidos como "*Timeout*" estão definidos não só para criar um intervalo de tempo para envio e gravação de variáveis, mas também para assegurar uma frequência para o ensaio constante, uma vez que o LCC não permite controlo sobre esta variável;

5. Envia por porta série o valor guardado na posição de memória W23, anteriormente descrito;

6. Salta para a macro seguinte, que neste caso é a macro número 11;

0		MacroNumber	11	
3	11-1	ForceMove	Target=-1000,Slope=	Negative direction force move at 122‰ of rated force
1	11-2	Wait	Time,Timeout=20	timeout=100
3	11-3	If	Variable,Var=Position_actual_value(0x006064),Lower,Var2=General_purpouse_r...	If actual position lower than W23 jump macro 15, else continu
3	11-4	SetOutput	Digital_output_0,On	Set digital output 0 to ON
1	11-5	GetVariable	Var=General_purpouse_registers-W50(0x322C00)	Read W50 register content
1	11-6	Wait	Time,Timeout=30	timeout=150
1	11-7	GetVariable	Var=Position_actual_value(0x006064)	Read position actual value
1	11-8	Wait	Time,Timeout=40	timeout=205
3	11-9	SetOutput	Digital_output_0,Off	Set digital output 0 to OFF
1	11-10	Macro	Jump,MacroNumber=12	

Figura 3.20: Macro número 11

A macro número 11, apresentada na Figura 3.20, consiste no controlo em força do ensaio e respetiva avaliação e comunicação, estando associada ao deslocamento retrativo da vara do atuador:

1. Desloca o atuador aplicando uma força de tração crescente no provete até esta atingir o valor em mira definido pelo utilizador. O atuador tem definido, pelo seu sistema de eixos, uma força de atuação positiva no sentido da sua extensão, pelo que para uma atuação de tração sobre o provete é necessário definir a força a atingir com um valor negativo;

2. Realiza um período de espera para estabilização do atuador;

3. Define uma condição de continuação de ensaio, apresentada na Figura 3.21. Se o valor de posição atual for inferior ao valor guardado na posição de memória W23, então o programa salta para a macro 15. Se a condição de finalização de ensaio não se realizar o programa prossegue na mesma macro;

4. Aciona a saída digital 0, que está ligada à placa Arduino;

5. O controlador envia por porta série a mensagem "D", que é o conteúdo do registo W50;

6. Realiza um período de espera para envio;

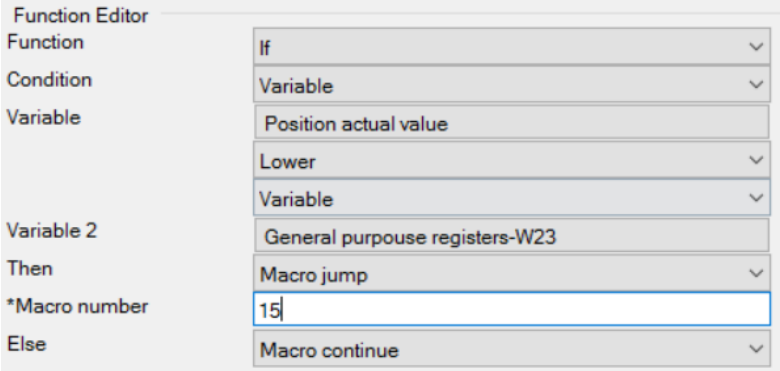

Figura 3.21: Função de final de ensaio por deslocamento

7. Obtém o novo valor de posição atual;
8. Realiza um período de espera para a obtenção do valor de posição atual e respetiva gravação;
9. Desliga a saída digital 0;
10. Salta para a macro número 12, que contém o resto do programa;

0		MacroNumber	12	
3	12-1	ForceMove	Target=-100,Slope=	Negative direction force move at 122‰ of rated force
1	12-2	Wait	Time,Timeout=20	timeout=100
3	12-3	SetOutput	Digital_output_0,On	Set digital output 0 to ON
1	12-4	GetVariable	Var=General_purpouse_registers-W45(0x2D2C00)	Read W45 register content
1	12-5	Wait	Time,Timeout=30	timeout=150
1	12-6	GetVariable	Var=Position_actual_value(0x006064)	Read position actual value
1	12-7	Wait	Time,Timeout=40	timeout=205
3	12-8	SetOutput	Digital_output_0,Off	Set digital output 0 to OFF
1	12-9	Macro	Jump,MacroNumber=13	

Figura 3.22: Macro número 12

Em semelhança com a macro descrita anteriormente (Macro 11), a macro 12 faz controlo em força do ensaio e respetiva avaliação e comunicação, estando esta associada ao deslocamento extensivo da vara do atuador:

1. Desloca o atuador aplicando uma força cujo valor foi definido de forma empírica, estando o valor associado à elasticidade do material, permitindo, juntamente com a inércia e peso do sistema, uma extensão do atuador. Uma vez que o provete permite flexão, a atuação de uma força positiva (i.e., de compressão) torna o sistema instável, como pode ser analisado e compreendido na Figura 3.23;
2. Realiza um período de espera para estabilização do atuador;

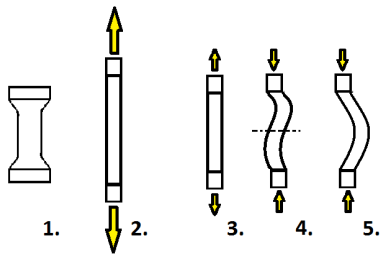

Figura 3.23: Comportamentos expectáveis de um provete (1.), à tração com uma força elevada (2.), com uma força inferior à sua elasticidade e peso da haste (3.), e à compressão com flexão (4. e 5.)

3. Aciona a saída digital 0;
4. Envia por porta série o valor da posição de memória do controlador W45, que contém o caracter "d";
5. Realiza um período de espera para obtenção da resposta ao requerimento anterior;
6. Guarda o valor de posição atual na variável "Var";
7. Realiza um período de espera para a obtenção do valor de posição atual e respetiva gravação;
8. Desliga a saída digital 0;
9. Salta para a macro número 13, que continua a execução do programa;

0		MacroNumber	13	
3	13-1	If	Variable,Var=General_purpouse_registers-W40(0x282C00),Higher,Const=499,Th...	If 500 iterations are reached, jump macro14, else continue ma
3	13-2	SetVariable	Var=General_purpouse_registers-W40(0x282C00),Add,Variable_and_constant,V...	Add 1 to W40 register content and store on W40
1	13-3	Macro	Jump,MacroNumber=11	

Figura 3.24: Macro número 13

Na macro 13 procede-se uma avaliação de número de ciclos do ensaio, cujo valor se encontra guardado na posição de memória do controlador W40, analisando se é superior ao número máximo de ciclos definido pelo utilizador para este registo. Se for verdade o programa irá chamar a marco número 14, caso contrário incrementa o valor de W40 e retorna à macro número 11 para repetição do ciclo;

A macro número 14, apresentada na Figura 3.25, tem como única função o incremento numa segunda variável. Uma vez que o número de ciclos máximos definidos pelo utilizador é 500 000, e não podendo este número ser incrementado apenas numa variável (W40), o que se faz e incrementar esta variável até 500, 1000 vezes. Assim o propósito desta macro é incrementar a variável W41 a cada 500 incrementos da variável W40, limpar o valor da última e retornar à macro número 11. Esta macro também avalia o fim

0		MacroNumber	14	
3	14-1	If	Variable,Var=General_purpouse_registers-W41(0x292C00),Higher,Const=999,Th...	If 500 iterations are reached, jump macro14, else continue ma
3	14-2	SetVariable	Var=General_purpouse_registers-W41(0x292C00),Add,Variable_and_constant,V...	Add 1 to W41 register content and store on W41
1	14-3	SetVariable	Var=General_purpouse_registers-W40(0x282C00),Constant,Const=0	Set W40 register content to zero
1	14-4	Macro	Jump,MacroNumber=11	

Figura 3.25: Macro número 14

de ensaio por chegada ao número máximo de ciclos, saltando para a macro 15 quando a avaliação retorna positiva.

0		MacroNumber	15	
2	15-1	Motor	Off	Turn motor OFF
1	15-2	GetVariable	Var=General_purpouse_registers-W100(0x642C00)	End program
1	15-3	Wait	Time,Timeout=20	
1	15-4	GetVariable	Var=General_purpouse_registers-W40(0x282C00)	End program
1	15-5	Wait	Time,Timeout=20	
1	15-6	GetVariable	Var=General_purpouse_registers-W41(0x292C00)	End program
3	15-7	SetOutput	Digital_output_0,Off	Set digital output 0 to OFF
3	15-8	SetOutput	Digital_output_1,On	Set digital output 1 to ON
1	15-9	Macro	Jump,MacroNumber=11	

Figura 3.26: Macro número 15

A macro número 15, apresentada na Figura 3.26, serve para concluir o ensaio, desligando o motor e terminando o programa. Como identificado anteriormente na análise das outras macros, o término do ensaio pode ser despoletado por um deslocamento superior a 2cm ou por superação do número máximo de 500 000 ciclos.

3.5.4 Obtenção de resultados

Terminado o ensaio, durante o qual o computador tem de estar permanentemente ligado e à escuta na sua porta série de comunicação com o *Arduino*, ficará gravado um ficheiro de texto com os valores recebidos, sendo necessário de seguida fazer a sua descodificação e análise. Para facilitar a explicação dos resultados, e tendo em conta que os ensaios podem durar dias e gerar ficheiros demasiado grandes para poder apresentar neste trabalho, apresenta-se só um excerto inicial do texto gerado durante um ensaio explicativo:

“Insert filename:
rFile name is:
r
Send s to stop
POS: 4294797784 AN: 1.36 ; 102.40 ; 0.50 Freq: 0.00
POS: 0 AN: 1.36 ; 102.40 ; 0.50 Freq: 5.56
POS: 205 AN: 1.36 ; 102.40 ; 0.50 Freq: 4.81
POS: 1 AN: 1.36 ; 102.40 ; 0.50 Freq: 5.15
POS: 206 AN: 1.36 ; 102.40 ; 0.50 Freq: 4.81

POS: 0 AN: 1.36 ; 102.40 ; 0.50 Freq: 5.10
POS: 206 AN: 1.36 ; 102.40 ; 0.50 Freq: 4.85
POS: 0 AN: 1.36 ; 102.40 ; 0.50 Freq: 5.15
POS: 207 AN: 1.36 ; 102.40 ; 0.50 Freq: 4.81
POS: 0 AN: 1.36 ; 102.40 ; 0.50 Freq: 5.15
POS: 207 AN: 1.36 ; 102.40 ; 0.50 Freq: 4.81"

Para a extração de informação podem utilizados vários softwares, tais como *Matlab*, *Excel*, etc., mas terão todos o mesmo tipo de abordagem ao problema, consistindo na criação de uma função que permita eliminar as primeiras 5 linhas, uma vez que não apresentam informação relevante, e que permita extrair os valores necessários de cada linha, que equivale a cada leitura pedida pelo controlador.

Terminada a extração de informação do ficheiro de texto gerado durante o ensaio cabe ao utilizador processar os dados de modo revelante para o estudo por ele realizado. Apesar do pós-processamento dos dados estar bastante dependente do estudo que está a ser desenvolvido pelo utilizador, existem dados revelantes e comuns a maioria dos casos, tais como forças máximas e mínimas, duração do ensaio, curvas tensão-deformação, etc., que podem ser extraídos de modo direto ou indireto dos dados obtidos e identificados pela aplicação desenvolvida.

Capítulo 4

Implementação de uma nova solução

Após a análise da máquina de ensaios existente e identificadas as suas capacidades e limitações, considerando os objetivos deste projeto, foi proposto um novo sistema, que posteriormente é desenvolvido e implementado, cujo diagrama básico de funcionamento pode ser observado na Figura 4.1.

Relembrando e resumindo os objetivos iniciais do trabalho, o projeto tem por base desenvolver uma solução modular simples, mais fácil e intuitiva de utilizar e com mais segurança e precisão na leitura dos dados.

Para responder a estas necessidades, foi desenvolvido o novo sistema, tendo por base os três módulos que compõem a máquina: o controlo, a leitura de dados e o sistema de interface com o utilizador.

O módulo de controlo e atuação sofreu poucas alterações, sendo que as mudanças principais se baseiam numa resposta a aumento de segurança da máquina, através da aplicação de um interruptor com proteção de picos de tensão e circuito de emergência.

No módulo de interface foi desenvolvido um novo *software*, com o intuito de facilitar a comunicação e interação da máquina com o utilizador, prescindindo da complexidade do sistema anterior e tendo uma abordagem mais gráfica e intuitiva como solução aos problemas antes identificados. Foi também utilizado um hardware diferente, com menor poder de processamento, mas mais flexível na utilização e adaptado à solução necessária.

O módulo de leitura foi completamente reestruturado, tendo sido alterado o hardware anteriormente utilizado de modo a produzir leituras mais precisas e fiáveis. Os novos equipamentos utilizados na leitura e comunicação de dados de teste apresentam também maior segurança contra interferências de ruídos e cargas externas e uma montagem mais compacta, mesmo assim permitindo a troca dos componentes principais para um menor tempo de manutenção em caso de avaria.

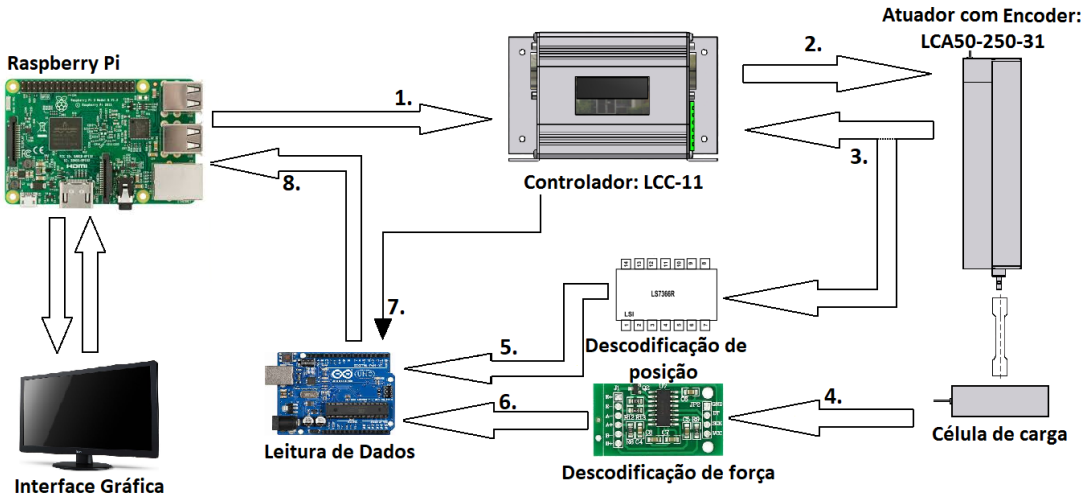

Figura 4.1: Diagrama de funcionamento da máquina de ensaios: 1.Programação do teste; 2.Controlo do atuador linear; 3.Leitura do encoder; 4.Leitura do sinal da célula de carga; 5.Leitura de posição; 6.Leitura de força; 7.Gatilho de envio de dados; 8.Envio de resultados;

4.1 Atuação e controlo

Uma das principais características e mais importantes duma máquina de ensaios é o seu método de atuação. Este fator altera as propriedades principais da máquina, como a força máxima exercida, deslocamento máximo e a velocidade que consegue atingir ou modificar a posição, entre outras, sendo estas propriedades determinantes no tipo de uso da máquina.

Um dos objetivos principais da máquina aqui descrita era e continuará a ser efetuar ensaios com possibilidade de grandes frequências e amplitudes de movimento. Como antes revisto, o atuador linear antes utilizado apresenta propriedades que permitem responder a essas necessidades sem a obrigação de instalação de sistemas auxiliares, permitindo criar uma máquina simples e ao mesmo tempo eficaz.

Devido à adequabilidade do equipamento antes utilizado e não se justificando um investimento noutro mecanismo cuja utilidade não iria suplantar o atual para os problemas que pretende resolver, não se alterou o módulo de atuação e controlo, permanecendo com a solução baseada no conjunto constituído pelo atuador LCA50-250-31 e pelo controlador LCC-11, já antes detalhadas.

Como referido, foi realizada uma atualização no sistema de segurança na alimentação do controlador, através da implementação de um disjuntor que fornece proteção contra picos de corrente da rede elétrica ao conversor de corrente alternada da rede para corrente contínua com 48V. Instalaram-se adicionalmente dois botões de segurança que interrompem a alimentação do controlador e consequentemente do atuador em caso de emergência. Um dos botões é um botão físico de emergência que permite interromper a alimentação de 48V de imediato quando pressionado. A outra interrupção consiste na integração de um relé conectado a uma saída digital do *Raspberry*, que interrompe

a alimentação quando um botão de emergência inserido na interface gráfica é pressionado. A dualidade de opções foi optada para permitir uma maior rapidez de resposta a emergência, estando o utilizador a interagir com o *software* ou com o *hardware*.

4.2 Obtenção de dados de teste

O módulo de obtenção de dados é um dos sistemas mais importantes da máquina, visto que o propósito da mesma é obter informação sobre a reação do material testado em esforços diferentes.

Na solução existente, estas leituras eram efetuadas com recurso a um *Arduino Mega* 2560, auxiliado por um descodificador dos sinais de encoder (HCTL 2032) e um circuito para retirar o ruído do sinal da célula de carga, sendo depois o sinal lido por uma porta analógica do microcontrolador.

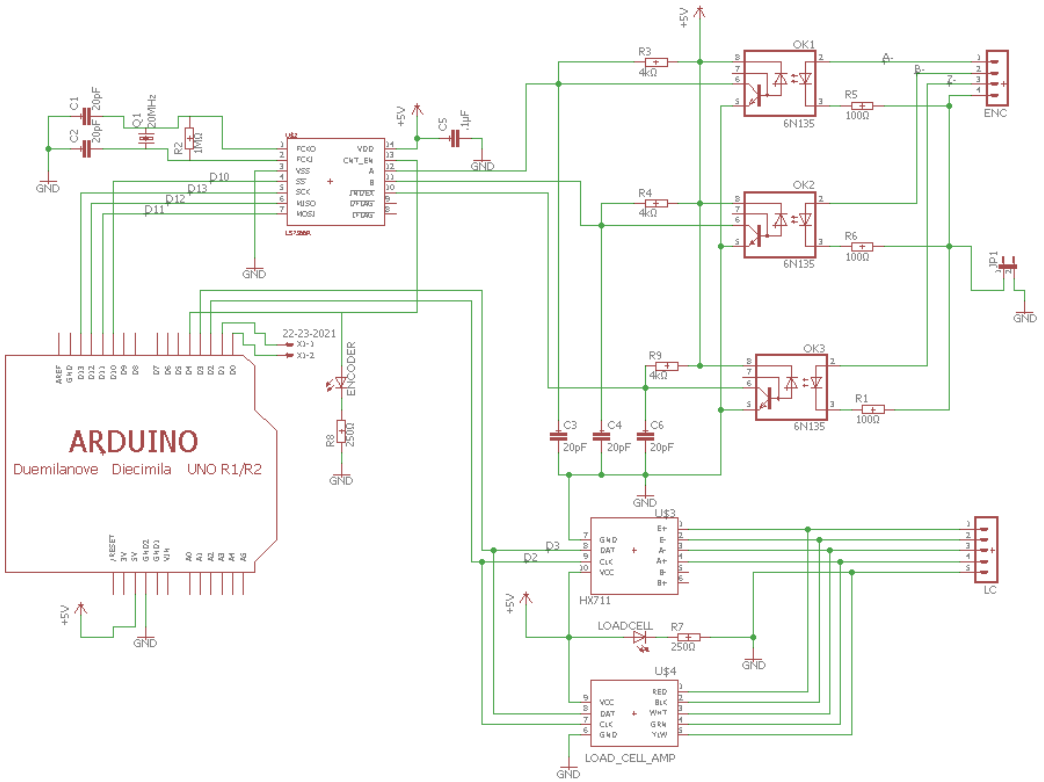

Figura 4.2: Esquema elétrico da placa de leitura de dados para o Arduino Uno

Para facilitar a leitura do valor de carga aplicada, foi utilizado um conversor de 24 bits de valores analógicos para digital (ADC[1]), desenvolvido especificamente para a interface com células de carga, cuja denominação no mercado é HX711 (34). Este

[1] "*Analog-to-Digital Converter*"

integrado torna mais simples e eficaz a leitura, fazendo a estabilização da alimentação da célula de carga, eliminando parte do possível erro logo de início e reduzindo assim alguma instabilidade da ponte de Wheatstone, fazendo também o tratamento do sinal proveniente e retirando o ruído. Como se pode observar na Figura 4.2, no quadrante inferior direito, recorreu-se ao uso de uma placa montada no circuito para proceder à obtenção dos sinais e à comunicação dos mesmos durante a leitura.

A utilização deste componente facilita também a calibração necessária quando se recorre a células de carga diferentes da utilizada para o projeto. Na solução anterior, uma vez que esta calibração se baseava numa análise e correspondência entre a carga aplicada e a tensão lida pelo microcontrolador, era necessário determinar uma curva de calibração para cada célula utilizada no programa, obrigando o utilizador a programar o módulo após cada calibração. Apesar da metodologia de calibração ser semelhante, sendo necessária a aplicação de uma carga conhecida por algum equipamento externo ao módulo, e leituras da mesma para obter uma relação entre ambas, o utilizador consegue agora ajustar o fator de calibração recorrendo à comunicação com o módulo, sendo esta já necessária para a leitura dos dados. A utilização do fator de calibração é intuitiva, aumentando o seu valor se o valor de força lido pelo módulo for inferior ao aplicado, e diminuindo se for superior. Após a calibração este valor fica guardado na memória não-volátil do microcontrolador, sendo lido na sua iniciação. A alteração do método de calibração retira a necessidade de programação do módulo, aumentando assim a sua flexibilidade de utilização.

Por uma questão de flexibilidade do sistema, desenhou-se o circuito de modo a aceitar duas placas de diferentes fornecedores, sendo as duas bastante semelhantes e tendo como componente principal o integrado HX711, acima referido, existindo apenas uma diferença na localização dos pinos de ligação. Após uma análise de desempenho das duas placas, ilustradas na Figura 4.3, concluiu-se que a placa verde (na figura à esquerda) é ligeiramente mais rápida e a roxa (na figura à direita) tem uma leitura mais estável.

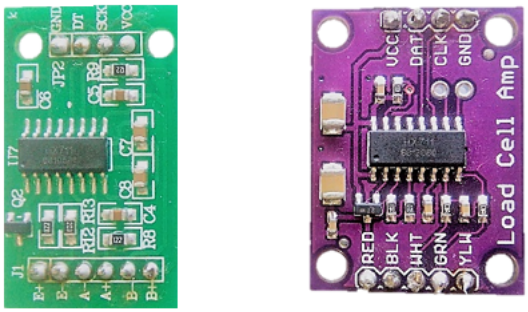

Figura 4.3: Placas de leitura da célula de carga baseadas no chip HX711

Também o componente de leitura de posição foi atualizado e simplificado, através da substituição do integrado HCTL-2032 por um LS7366R (35). Apesar de ambos os dispositivos serem descodificadores de quadratura e operarem com precisões semelhantes, apresentam grande diferença na comunicação de resultados com o microcontrolador a que estão acoplados. Enquanto que o antigo componente (HCTL-2032) comunica com

o *Arduino* através de uma ligação em paralelo utilizando 16 portas, o novo componente comunica com o microcontrolador recorrendo ao protocolo de comunicação SPI[2].

Para aumentar a segurança no sistema e isolar o funcionamento do *Arduino* e da placa com o do encoder e atuador, protegendo ambos os circuitos de picos de tensão, recorreu-se ao uso de optoacopladores de alta velocidade. Foi utilizado um acoplador ótico baseado no foto díodo 6N135 (36) para os sinais $A-$, $B-$ e $Z-$ provenientes do encoder, utilizando o mesmo método de captação de sinal que o utilizado na versão anterior da máquina. Este dispositivo inverte o sinal recebido, devolvendo ao integrado de leitura de posição os sinais $A+$, $B+$ e $Z+$, podendo transferir o sinal com frequências até 2 MHz.

Através das simplificações descritas neste capítulo, foi possível simplificar também o microcontrolador necessário que serve de base ao módulo de leitura, reduzindo o número de portas necessárias e reduzindo o poder de cálculo necessário do mesmo. Explorando essa simplicidade acrescida, substituiu-se o *Arduino Mega* por um *Arduino Uno*, cujas características estão descritas na Figura 4.4.

Name	Processor	Operating/Input Voltage	CPU Speed	Analog In/Out	Digital IO/PWM	EEPROM [kB]	SRAM [kB]	Flash [kB]	USB	UART
Uno	ATmega328P	5 V / 7-12 V	16 MHz	6/0	14/6	1	2	32	Regular	1
Mega 2560	ATmega2560	5 V / 7-12 V	16 MHz	16/0	54/15	4	8	256	Regular	4

Figura 4.4: Comparação de especificações entre o *Arduino Mega* e o *Arduino Uno* (13)

Para tornar a solução mais compacta e manuseável, e tendo sido decidida a utilização do *Arduino Uno* como microcontrolador de base para o módulo de recolha, leitura e comunicação de dados, procedeu-se à realização de uma placa de circuitos impresso, onde todos os componentes estão montados, tendo por base o circuito mostrado na Figura 4.2. A placa projetada e construída tem uma dimensão aproximada de uma shield convencional de *Arduino Uno*, apresentada na Figura 4.5,na qual os componentes principais (integrado de leitura de força e posição e optoacopladores), assim como as ligações a outros componentes (conectores para os sinais provenientes do encoder e célula de carga e comunicação) foram acoplados de forma a serem facilmente substituíveis quando necessário.

4.3 Interface e gravação de resultados

A interface do utilizador constituiu uma das principais motivações para a realização deste trabalho e para a substituição do sistema anterior. Apesar de igual relevância para o projeto, este módulo tem mais impacto no utilizador da máquina, uma vez que permite a interação com os dispositivos digitais e com o equipamento como um todo e permite a correta e devida utilização do mesmo. O objetivo da interface gráfica para o utilizador

[2] "*Serial Peripheral Interface*"

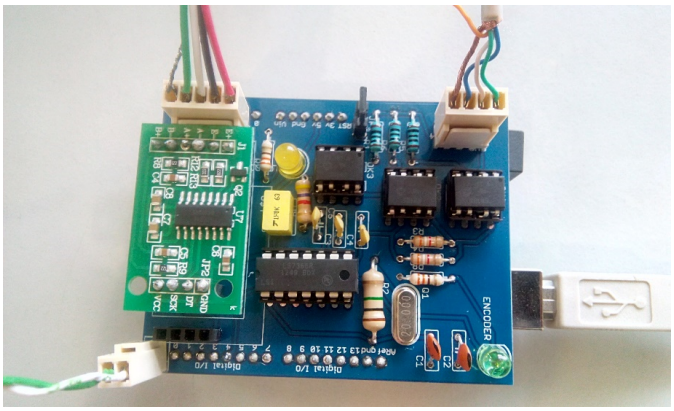

Figura 4.5: Placa de circuito impresso desenvolvida para o módulo de leitura de dados

é permitir a interação entre utilizador e a máquina recorrendo a elementos gráficos, um rato e/ou teclado, ou ecrã tátil, com os quais o usuário é capaz de selecionar símbolos e inserir texto e manipulá-los de forma a corretamente e facilmente programar e efetuar um teste material.

Como antes analisado, a solução antiga baseava a sua interface no programa "*SMAC Control Center*", disponibilizado pelo fabricante do atuador e controlador, auxiliada por um programa de leitura da porta série para receber e gravar resultados, estando ambas aplicações a correr num computador fixo dedicado à máquina. Esta solução, para além de pouco prática pela constante necessidade de mudança de aplicação, produz uma interface de difícil assimilação para um utilizador inexperiente, requerendo capacidades de programação; tendo sido desenvolvida para a programação pontual de um atuador em ambiente fabril e não para uso diário para a realização de testes que envolvam várias mudanças de variáveis.

Devido à necessidade de dedicação do hardware à máquina, e sendo o computador utilizado na solução antiga um recurso poderoso, recorreu-se ao uso de um equipamento mais pequeno, barato, mas mais versátil também na prototipagem e programação de equipamentos. Neste sentido, o computador antes utilizado foi substituído por uma plataforma de desenvolvimento *Raspberry Pi*, que é um minicomputador com 40 pinos de entradas e saídas programáveis, 4 portas USB, saída de vídeo HDMI e possibilidade de ligação por WIFI e *Bluetooth*, sendo esta solução capaz de suprir as necessidades da máquina de ensaios.

Foram analisadas outras hipóteses antes de se ter tomada a decisão de recorrer ao *Raspberry Pi*, tendo algumas sido inclusivamente testadas e desenvolvidas até se ter concluído que não podiam ser utilizadas. Uma das primeiras soluções analisadas consistia no uso de um *Arduino Uno*, completado por uma *shield WIFI CC3000* e um ecrã *TFT LCD de 3.5"*. Esta solução foi descartada pela incompatibilidade de uso das duas *shield* no mesmo microcontrolador. Para possibilitar a utilização do ecrã tátil, o *Arduino Uno* e a respetiva *shield WIFI* foram substituídos por uma placa *ESPDUINO*. Este equipamento consiste na integração de um microcontrolador *ESP-13* numa placa semelhante à do *Arduino Uno*, permitindo assim a montagem de *shield* específicas ao último. Verificou-se

Figura 4.6: *Raspberry Pi* 3, modelo B (14)

que as conexões disponíveis da placa não eram compatíveis com a utilização do ecrã tátil e verificou-se que devido à interação necessária com o utilizador e devido ao tamanho reduzido do componente seria mais relevante aproveitar um *display* de computador.

De seguida ponderou-se a utilização de um *ESP8266* para gerir as comunicações e interagir com uma base de dados, cuja informação permitia realizar a interface através de qualquer dispositivo remoto com acesso ao servidor. Apesar desta solução ser tecnologicamente mais avançada e permitir a interação com a máquina através de um computador não dedicado, *tablet* ou telemóvel, não apresenta benefícios suficientes para a complexidade que acrescenta, pois para a montagem do provete de ensaios é necessária a presença do utilizador junto à máquina. Esta solução oferecia as vantagens de monitorização remota e avisos de fim de ensaio, mas acrescentaria a dificuldade de integração de interface nas várias plataformas de forma a manter a sua utilidade.

O equipamento escolhido, além de permitir uma incorporação bastante simples de ecrã, teclado e rato para a facilitar a interação com o utilizador, possui também a opção de utilização de WIFI, permitindo um desenvolvimento futuro de um programa que possibilite realizar quer o envio de dados de ensaio para monitorização, quer o aviso de fim de ensaio, como o envio de um relatório padrão preliminar para o email do utilizador. A disponibilidade de GPIO integrados no equipamento permite também uma maior flexibilidade de utilização e uma mais fácil integração do módulo.

Uma vez completa a seleção de hardware e os outros módulos estarem definidos, foi necessário definir o ambiente de programação de teste. Por uma questão de universalidade do produto e compreensão mais generalizada das funções dos botões e etiquetas, a interface gráfica desenvolvida foi desenhada em língua inglesa.

Durante a inicialização do controlador é chamada a função de busca do sinal indicador da posição zero, que provoca uma série de movimentos bruscos e repentinos no atuador. Estes movimentos podem danificar o equipamento, principalmente se um provete já estiver montado no mesmo, e podem tornar-se perigosos para o utilizador ou qualquer outra pessoa perto da máquina, pelo que a aplicação desenvolvida, ao iniciar, apresenta uma mensagem de aviso visível na Figura 4.7, obrigando o utente a verificar antes de confirmar o início do equipamento. Ao premir no botão de iniciar a máquina (na Figura 4.7 o botão "*Start the Machine*"), recorrendo a um relé compatível e a uma saída digital do *Raspberry*, o controlador é iniciado e os movimentos vão ser possibilitados.

Figura 4.7: Janela inicial do programa: Janela de iniciação do atuador e controlador

Após iniciado o controlador, é possível comunicar com ele, permitindo assim o movimento da haste móvel do atuador através dos botões situados do lado direito do ambiente principal de programação de teste, zona identificada na Figura 4.8 com a etiqueta "*Movement Buttons*". Nesta zona estão disponíveis os botões "*Start*" e "*Stop*", que ativam ou desativam o relé de alimentação do controlador, respetivamente, possibilitando assim uma paragem de emergência ou uma iniciação do equipamento após assegurar as condições para a sua realização. Estão também representados na Figura 4.8 os botões de movimento manual, identificados com setas, identificando o sentido de movimento do atuador e o deslocamento, sendo que os botões que figuram as duas setas representam um deslocamento maior, de 3cm e os que apenas têm uma seta representam um deslocamento de apenas 5mm. Recorrendo a estes movimentos é então possível a correta colocação do provete na máquina para a realização do teste requerido. O provete só deve ser montado nos seus suportes após a inicialização do atuador e do seu correto posicionamento.

À semelhança da solução anterior, o programa grava os resultados num ficheiro de texto, passível de ser lido no final do ensaio para tratamento dos dados e obtenção dos resultados. Este ficheiro é gerado automaticamente aquando a gravação do teste no controlador, sendo para isso necessário o nome do utilizador a requisitar o ensaio e o nome do ficheiro, criando assim um ficheiro de texto (.txt) cujo nome inclui também a data do ensaio. Neste ficheiro ficam guardados todos os dados para posteriormente se poder identificar quem requisitou o teste e poder repetir o mesmo, tendo guardados o tipo e parâmetros de teste, assim como condições do fim do mesmo.

Na janela de programação de teste, na zona central do ambiente visualizado na Figura 4.8, é possível ainda de visualizar os valores de força e de posição atuais do atuador. Este valor é atualizado com a gravação dos dados no ficheiro de texto, sendo por isso possível a visualização e controlo em tempo real dos resultados do ensaio, e verificação da conformidade com o expectável, sendo possível terminar o teste precocemente através do botão situado na zona central inferior identificado como "*Stop Test*".

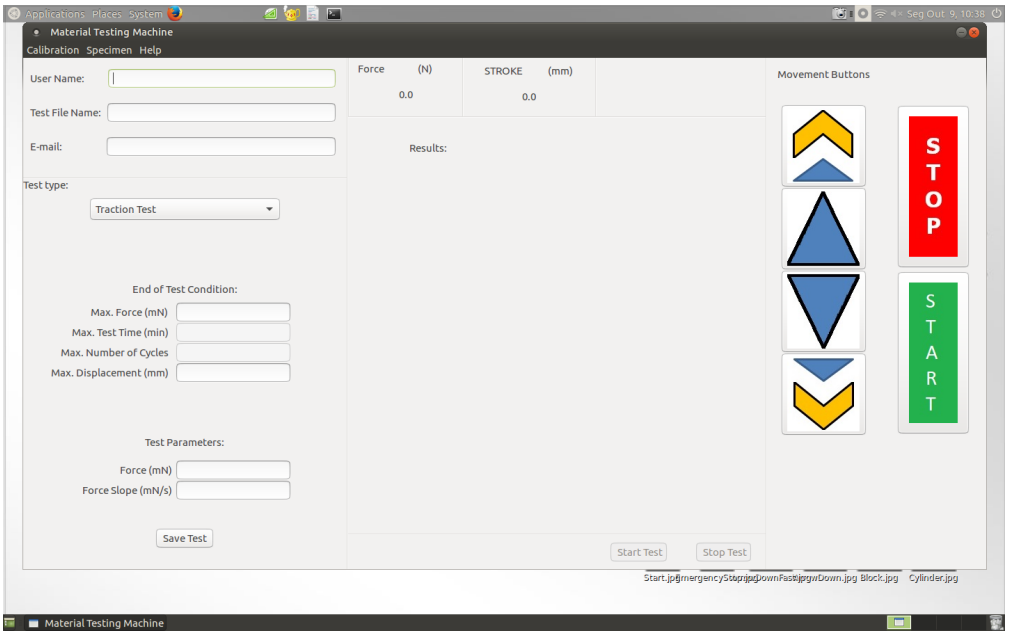

Figura 4.8: Janela principal de programação de testes e movimentos do atuador

Após já se ter analisado como se procede para iniciar o equipamento, quando se deve realizar o aperto do provete e como são guardados os dados, resta compreender a dinâmica de programação do ensaio propriamente dito.

Como se pode observar pela Figura 4.9, a aplicação desenvolvida permite programar três tipos de teste padrão: Ensaios de tração/compressão, ensaios de flexão com 3 ou 4 apoios e ensaios cíclicos, sendo estes últimos dois tipos de ensaios programáveis por imposição de força ou de deslocamento, enquanto que os ensaios de tração e compressão apenas atuam por força. Na seleção de um tipo diferente de ensaio os parâmetros de teste, visíveis na Figura 4.8 no canto inferior esquerdo do ambiente, vão se alterar para corresponder aos necessários para a elaboração do teste, sendo necessário o preenchimento de todos para o que utilizador obtenha a permissão para guardar o programa de ensaio no controlador, sendo necessário também o preenchimento de pelo menos uma condição de fim de teste.

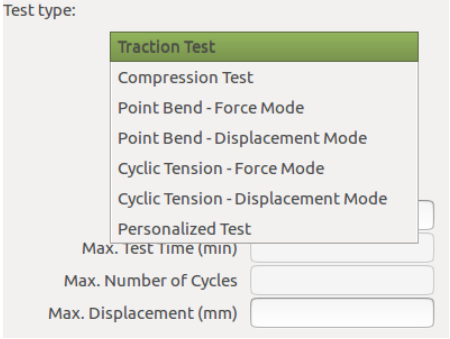

Figura 4.9: Escolha de tipo de teste material a realizar

Na programação de ensaios estão disponíveis quatro condições de fim de teste, como a força máxima a atuar e deslocamento máximo do atuador, ambas comuns a todos os ensaios e foram projetadas principalmente para garantir a salvaguarda do equipamento. Para os ensaios que envolvem movimentos repetidos, sendo de flexão ou cíclicos, é possível definir como condição o número máximo de ciclos a exercer antes da paragem ou o tempo máximo de teste. Para o caso de o campo de condição não estar preenchido, a programação do ensaio realiza-se com um valor atribuído pelo código, estando a força máxima definida como 90 N, sendo essa a força máxima exercida pelo atuador utilizado, o deslocamento máximo definido como 10 cm, para possibilitar a paragem do ensaio devido a rutura, e o número de ciclos máximo de 5×10^7 repetições, sendo este valor propositadamente um valor elevado, de forma a não parar um ensaio antecipadamente.

Os parâmetros que são necessários para cada ensaio são variáveis, como é demonstrado na tabela 4.1, sendo por isso necessário compreender o que significa cada um deles:

- Força ("*Force*"): Introduzido em mN, é um parâmetro dos ensaios de tração e de compressão. Representa o incremento de força por segundo que se pretende efetuar, até à força máxima definida pela condição de fim de teste ou até à rotura do provete;
- Força Superior ("*Force High-Point*"): introduzido em mN, é um parâmetro dos ensaios cíclicos que atuam por controlo direto da força. Representa a força superior a aplicar sobre o provete durante o ciclo de atuação;
- Força Inferior ("*Force Low-Point*"): introduzido em mN, é um parâmetro dos ensaios cíclicos que atuam por controlo direto da força. Representa a força inferior a aplicar sobre o provete durante o ciclo de atuação. Em ensaios que se pretenda fazer ciclos de carga é recomendado que se introduza uma força próxima, mas diferente de 0 N, para evitar a flexão do provete quando o fenómeno não é desejado;
- Declive de força ("*Force Slope*"): Introduzido em mN/s, é o parâmetro que indica a velocidade de atuação da força. O mau preenchimento deste parâmetro pode significar o incumprimento dos parâmetros de aplicação de força, isto é, o controlador passar a um comando diferente sem a força ter sido atuada na totalidade. Obrigar o controlador a esperar pela execução do comando significa um incumprimento da frequência requerida ao longo do ensaio todo;
- Deslocamento ("*Displacement*"): Introduzido em mm. É um parâmetro dos ensaios cíclicos em que a atuação se baseia no deslocamento. Representa a variação de posição do atuador durante o ciclo, a partir da posição inicial do mesmo;
- Velocidade ("*Velocity*"): Introduzido em mm/s, é igualmente um parâmetro dos ensaios cíclicos por deslocamento, e cujo modo de atuação no programa é semelhante ao declive da força, representando o declive do deslocamento;
- Frequência ("*Frequency*"): Introduzido em ciclos/s ou Hz, é um parâmetro dos ensaios cíclicos. Este parâmetro permite calcular o intervalo de espera a efetuar pelo controlador entre comandos de aplicação de força ou deslocamento do atuador. Valores demasiado elevados de frequência, para além de terem menor precisão

Tabela 4.1: Parâmetros relativos a cada tipo de ensaio

Teste/Parâmetros	Força	Força Superior	Força Inferior	Declive de força	Deslocamento	Velocidade	Frequência
Tração	■			■			
Compressão	■			■			
Flexão (Força)		■	■	■			■
Flexão (Deslocamento)					■	■	■
Força Cíclica		■	■	■			■
Deslocamento Cíclico					■	■	■

garantida, podem significar também uma aplicação incompleta da força ou deslocamento;

O programa possui ainda uma janela que possibilita a definição de um provete, como ilustrado na Figura 4.10. O objetivo desta janela é guardar os dados relativos à geometria do provete no ficheiro de texto, para auxiliar o utilizador no tratamento de dados posterior ao ensaio, permitindo uma rápida obtenção dos valores, prevenindo a invalidez do tratamento por esquecimento dos mesmos, ou mesmo possibilitando ao utilizador a realização de uma série de ensaios com provetes de diversas geometrias e/ou tamanhos com maior facilidade e organização.

Como se observa na Figura 4.11, é possível editar seis tipos de provetes, sendo eles o provete tipo haltere, retangular, tipo arame, bloco, cilíndrico ou viga. Mudando a geometria através da seleção na caixa de combinações, é representada uma imagem na zona direita da janela ilustrativa do provete e das suas dimensões relevantes, sendo a sua determinação realizada nas caixas identificadas com a mesma nomenclatura na zona esquerda. Idealmente, todas as medidas deverão ser inseridas em mm, para consistência na escrita e leitura e para não limitar as vantagens acima referidas.

A definição da geometria do provete e das suas medidas não é requerida para a realização do ensaio, não havendo, portanto, avisos relativamente à falta destes dados. Todavia, pelas razões acima referidas e por poder ser considerada uma mais valia para um utilizador que tenha de lidar com múltiplos ensaios, a introdução desta informação é recomendada.

4.4 Comunicação e agregação de hardware: Protocolos

Para possibilitar a programação de ensaios numa máquina constituída por componentes modulares é necessário estruturar uma base de comunicação sólida, permitindo o bom funcionamento da máquina como um todo ou perante a substituição de um módulo. Além desta comunicação entre módulos constituintes da máquina, é de notar que existem comunicações cuja existência é fundamental para o seu desempenho na tarefa

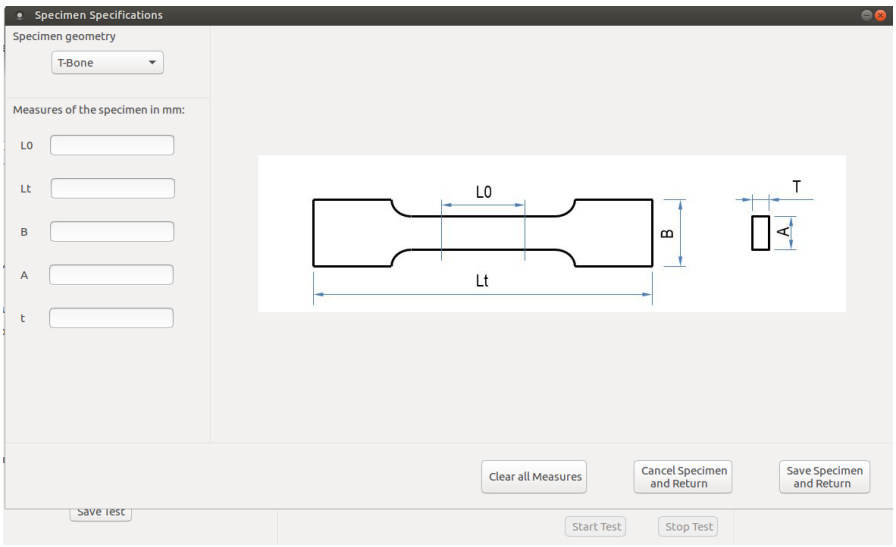

Figura 4.10: Janela de edição de geometria do provete

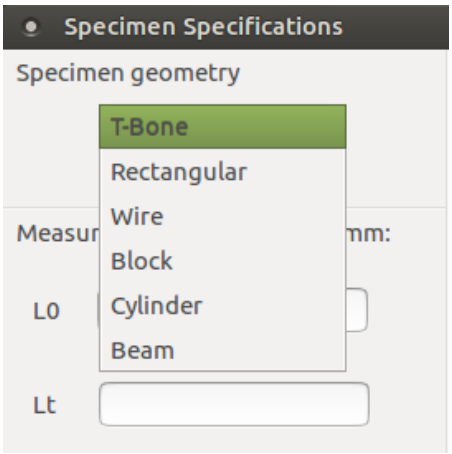

Figura 4.11: Opções de geometria do provete

requerida. Porém parte destas comunicações já foram exploradas na análise dos módulos acima descritas, nomeadamente a comunicação necessária no módulo de leitura de dados entre os componentes de leitura de posição e carga e o seu controlador agregador.

4.4.1 Comunicação Arduíno-Interface

Este módulo envia a mensagem cujo conteúdo são os dados lidos do teste aquando a deteção de um sinal de *trigger*, quer seja a receção do caracter "s", alusivo ao comando "*send*", pela porta série, quer seja a deteção de sinal na entrada digital dedicada, permitindo assim o pedido de leitura aos outros dois módulos. Embora o evento principal para pedido de leitura de dados durante o ensaio seja iniciado pelo controlador, através do acionamento de uma saída digital programada nas macros dos ensaios, a capacidade de pedido de leitura por parte do módulo de interface acrescenta flexibilidade ao sistema, permitindo leituras isoladas dos valores de posição e força.

Esta mensagem é constituída pela indicação da força aplicada, em mN, seguida da leitura de posição, em nm. Apesar de na escala de medida a leitura de carga apresentar alguma interferência e a resolução do encoder ser de 1μm, esta escala foi utilizada para permitir ao utilizador fazer a deteção e análise de erro na leitura.

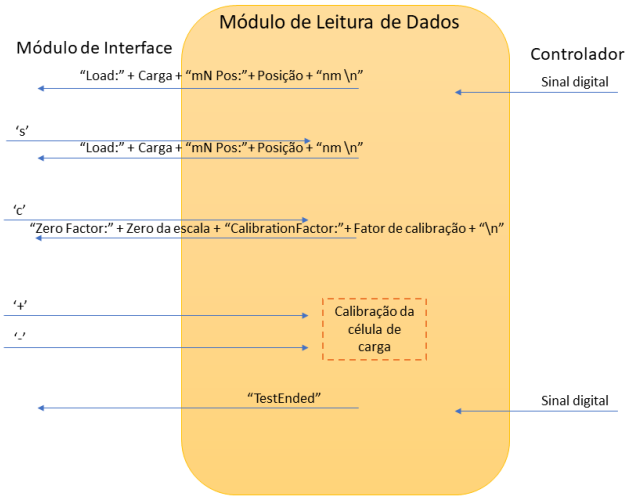

Figura 4.12: Estrutura da comunicação com o módulo de leitura de dados

Como se pode observar na Figura 4.12, foram também definidos protocolos de comunicação que possibilitam a calibração de uma célula de carga diferente, aumentando a flexibilidade do módulo no tipo de ensaios possíveis de realizar. A calibração da célula de carga deverá ser realizada recorrendo a outro equipamento já calibrado, que aplica uma força conhecida. Através da leitura da carga é possível ir ajustando o fator de calibração, através do envio dos caracteres "+" e "-", até as medições serem coerentes com as forças exercidas. Após a calibração estar realizada deve-se fazer a leitura do fator de calibração que permite as medições mais precisas, ficando esse valor guardado

na memória não-volátil (*EEPROM*) do *Arduino*. Essa leitura é requerida através do envio do caractere "c", alusivo à palavra "*calibration*".

Devida à simplicidade na comunicação e da construção deste módulo, torna-se possível de utiliza-lo noutros projetos que necessitem da leitura de valores de posição e de carga, sendo os equipamentos compatíveis de uso corrente e comuns num ambiente industrial e universitário.

4.4.2 Comunicação Interface-Controlador

Sendo a máquina constituída por módulos que assumem diferentes funções dentro da mesma, apresenta-se uma separação entre o utilizador que comanda o atuador e o equipamento que o controla.

A ligação entre os dois módulos baseia-se na mesma arquitetura de comunicação que o programa utilizado na versão anterior da máquina, sendo este o "*SMAC Control Center*". É de notar que, como antes referido, o processo de interface não comunica diretamente com os módulos, auxiliando-se de um processo simultâneo que gere as comunicações e parte das saídas digitais.

O processo de comunicação com o controlador foi simplificado, envolvendo principalmente a chamada e a gravação de macros, como anteriormente, mas retirando do utilizador toda a necessidade de programação nos ensaios definidos. O controlador possuí agora macros fixas e macros que são alteradas para acomodar os parâmetros do teste. Cada teste tem macros específicas e dedicadas, estando elas identificadas:

0- Iniciação do atuador- Realiza a deteção de fase no atuador, procura a sua posição inicial e de seguida move-se até uma posição estipulada;

1- Movimento rápido e longo descendente- move o atuador 10 mm;

2- Movimento rápido e longo ascendente- move o atuador 10 mm;

3- Movimento lento e curto descendente- move o atuador 1 mm;

4- Movimento lento e curto ascendente- move o atuador 1 mm;

10- Início do teste de tração/compressão por incremento de força- Define as condições de fim de teste e chama a Macro 11;

11- Continuação do teste de tração/compressão por incremento de força- Incrementa a força atuada, analisa a veracidade das condições de fim de teste, envia pedidos de leitura de dados ao módulo dedicado e repete até uma das condições de fim ser realizada;

15- Início do teste de tração/compressão por incremento de posição- Define as condições de fim de teste e chama a Macro 16;

16- Continuação do teste de tração/compressão por incremento de posição- Incrementa a posição, analisa a veracidade das condições de fim de teste, envia pedidos de leitura de dados ao módulo dedicado e repete até uma das condições de fim ser realizada;

20- Início dos testes de ensaios cíclicos de atuação por força- Define a condição de fim de teste por extensão máxima e faz *reset* dos contadores de número de ciclos, de seguida chama a Macro 21;

21- Continuação do teste cíclico por força- Atua a rampa de subida e envia pedido de leitura;

22- Continuação do teste cíclico por força- Atua a rampa de descida e envia pedido de leitura;

23- Continuação do teste cíclico por força- Analisa o fim de teste por deformação e incrementa o número de ciclos;

24- Continuação do teste cíclico por força- Analisa o fim de teste por número de ciclos;

25- Início dos testes de ensaios cíclicos de atuação por deslocamento- Faz *reset* dos contadores de número de ciclos, de seguida chama a Macro 26;

26- Continuação do teste cíclico por deslocamento- Atua a rampa de subida e envia pedido de leitura;

27- Continuação do teste cíclico por deslocamento - Atua a rampa de descida e envia pedido de leitura;

28- Continuação do teste cíclico por deslocamento - Analisa o fim de teste por força máxima e incrementa o número de ciclos;

29- Continuação do teste cíclico por força- Analisa o fim de teste por número de ciclos;

50- Macro de fim de teste- Desliga o atuador e envia sinal de fim de teste;

Macros fixas foram definidas como sendo as macros que o programa não tem necessidade de alterar para o seu correto funcionamento, sendo elas macros de movimento simples (1 a 4), para o posicionamento do atuador e para montagem de provete, macros que não envolvam variáveis (23, 25 e 28) e as macros de iniciação do atuador (0) e fim de teste (50). Estas macros são gravadas por uma questão de segurança após a iniciação do controlador. As macros cujo conteúdo é variável devido a alteração dos parâmetros de teste têm de ser gravadas na íntegra a cada ensaio novo, sendo só gravadas as macros envolvidas no mesmo.

O equipamento utilizado para a gestão da interface gráfica e comunicação com os módulos de controlo e leitura, o *Raspberry Pi*, apenas possui uma saída *UART*, que é utilizada na comunicação com o módulo de leitura, pelo que se recorreu a uma biblioteca para realizar uma técnica de "*Bit Banging*" para comunicar com o controlador. Esta técnica consiste na utilização de pinos não dedicados à comunicação série para realizar a mesma, efetuando o controlo de sinal através de software. Esta técnica, no caso específico do equipamento utilizado, apresenta a limitação de possuir uma taxa de transmissão máxima inferior aos 115200 bps descritos anteriormente com sendo o *baudrate* do controlador, pelo que se definiu a nova velocidade de transmissão de bits como sendo 9600 bps. Apesar da alteração apresentar maior estabilidade de comunicação, a

alteração implica que o controlador já não consegue comunicar com o programa "*Smac Control Center*". A decisão de utilizar os GPIO do equipamento em vez de recorrer ás portas USB deveu-se a reservar estas portas para integração do teclado e rato, deixando ainda duas portas disponíveis para a inserção de uma pen drive para permitir a recolha dos resultados dos testes.

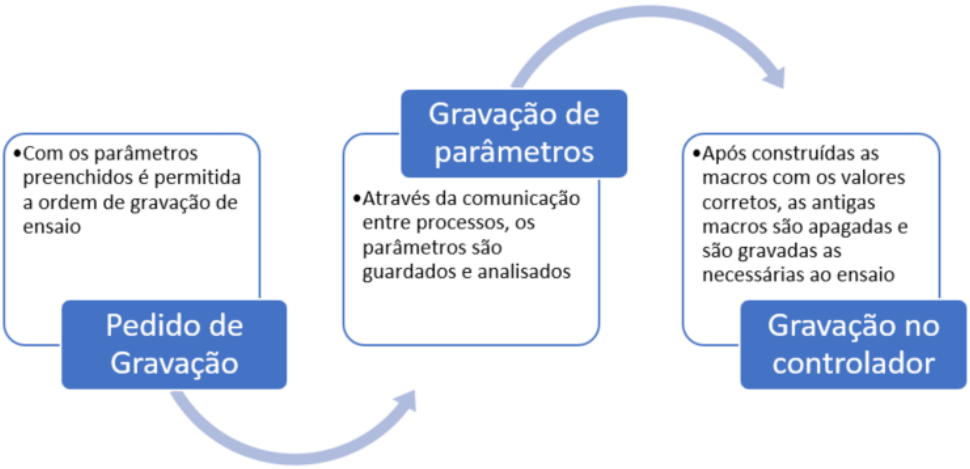

Figura 4.13: Fluxo de comunicação na gravação de ensaios

Como se pode observar pela Figura 4.13, durante o pedido de gravação de programa, o processo de interface irá comunicar o valor de todos os parâmetros necessários, incluindo o tipo de teste. Através do tipo de ensaio, o processo responsável pela comunicação com os outros módulos deteta quais as macros que deve enviar, procedendo antes à sua modificação para aceitar os parâmetros requeridos pelo utilizador, convertendo as suas unidades para as utilizadas pelo controlador. Após a gravação do ensaio é possível iniciá-lo pedindo a chamada da sua macro inicial. Na chamada da macro o programa envia o código de fim de programa ("0x20 W 0x032C02 0"), seguido do código de chamada da macro ("0x20 W 0x012C04 "+ Nº da Macro). Estes códigos, tais como os de gravação de macro, são específicos da biblioteca SMAC e foram previamente implementados no controlador.

4.4.3 Comunicação entre processos no módulo Interface

Como antes referido, o módulo de interface gráfica com o utilizador funciona como um módulo agregador do conjunto, necessitando de comunicar com os outros módulos para fornecer ao utilizador a leitura dos dados e providenciar o controlo requerido pelo mesmo na atuação de um ensaio.

Dada a necessidade de monitorização constante, quer de interação com o utilizador, quer de informação recolhida, foi necessário separar o módulo constituído pelo *Raspberry Pi* em dois processos distintos, embora dependentes, para gerir estas duas necessidades. A separação das funções, embora permita uma maior especialização de cada processo, origina uma necessidade de comunicação interna.

A comunicação entre processos (IPC- "*Inter-Process Communication*"), realiza-se mediante a transferência de uma mensagem padrão, interpretável por ambos os processos, na memória partilhada pelos mesmos. A mensagem é escrita na memória partilhada, quando o processo requerente envia um sinal ao processo destinatário do pedido. Após a leitura da mensagem, é enviada uma resposta de descodificação da mesma, permitindo ao utilizador ter conhecimento da mesma e identificar possíveis erros de comunicação através da visualização de uma mensagem relativa ao mesmo na zona inferior central da janela principal, como se observa na Figura 4.8.

As mensagens escritas na memória partilhada usam um formato único, sendo utilizadas principalmente para a escrita de valor em variáveis ou confirmação da ação prévia. Cada mensagem consiste na concatenação da variável que se pretende modificar, com o valor ou texto que se pretende associar, estando estes dois objetos divididos pelo carater de traço inferior "_". A utilização deste símbolo é requerida apenas para simplificar a descodificação da mensagem, facilitando a separação da cadeia de texto.

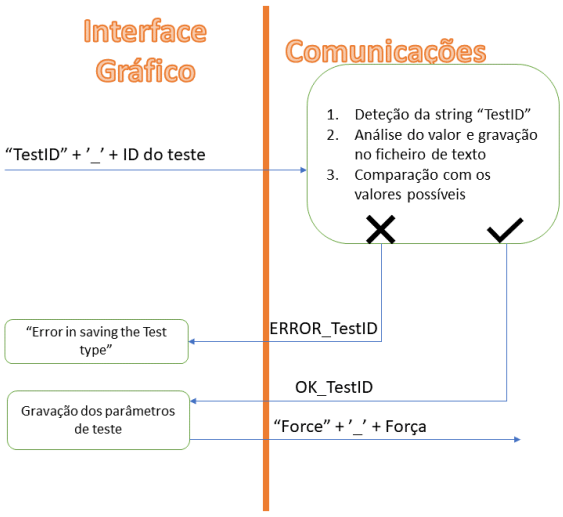

Figura 4.14: Exemplo de sequência de mensagens para gravação de um parâmetro

Na Figura 4.14 pode observar-se a sequência de mensagens envolvida na gravação de um parâmetro, neste caso do tipo de ensaio a realizar. O envio da mensagem é acionado pelo pedido do utilizador para a gravação no controlador do ensaio requerido com os parâmetros introduzidos. Após a escrita na memória partilhada, o processo responsável pela interface gráfica envia o sinal *SIGUSR2*, que é detetado pelo processo responsável pelas comunicações, e que irá posteriormente ler o conteúdo da memória partilhada e procurar pela palavra identificadora do parâmetro, neste caso "*TestID*". Este parâmetro aceita números identificadores de cada tipo de teste, pelo que vai analisar se o conteúdo da mensagem após o traço inferior é um número natural pertencente ao conjunto aceitável. Caso esta condição esteja garantida, o tipo de teste é definido conforme o número recebido, e é escrito no ficheiro de texto dos resultados, sendo de seguida enviada a mensagem de confirmação de descodificação da mensagem, como se pode observar na figura 4.14. Caso contrário, quer por o valor recebido não ser numérico,

quer por não pertencer ao intervalo aceitável, é enviado para o processo responsável pela interface a mensagem de erro na gravação do tipo de teste.

4.5 Análise de um teste

Com o intuito de fazer o paralelismo entre a análise da solução anterior com a solução agora implementada, vai proceder-se à análise de um exemplo de ensaio cíclico, apresentando os passos para a sua execução e respetiva análise.

Estando as ferramentas necessárias para a execução e respetiva obtenção e tratamento de dados dos ensaios de testes materiais agrupadas num só programa, é apenas necessário iniciar o programa desenvolvido, clicando no ícone presente no ambiente de trabalho do *Raspberry Pi*, que pode ser observado na Figura 4.15.

Após duplo *click* no ícone da máquina de ensaios é apresentada ao utilizador uma janela em que é pedida a introdução da palavra passe do *Raspberry Pi*, de modo a fornecer à aplicação permissão de administrador e possibilitar assim o acesso aos GPIO's.

Figura 4.15: *Desktop* do *Raspberry Pi*, com o ícone de início do programa no canto superior esquerdo

Depois de iniciada a aplicação, é aberta uma janela que possibilita a inicialização do controlador, advertindo para a precaução a ter para manter a segurança perto do equipamento.

Pressionando no botão "*Start the Machine*", presente na janela e observável na Figura 4.16, o controlador, atuador e módulo de leitura serão iniciados, sendo fornecida alimentação elétrica aos mesmos acionando um par de relés através dos pinos do *Raspberry Pi*. Um dos relés controla o fornecimento de alimentação ao circuito de 48V (Controlador e

Atuador), enquanto que o outro permite a passagem de 5V para alimentar o Arduino e todo o módulo de leitura de dados.

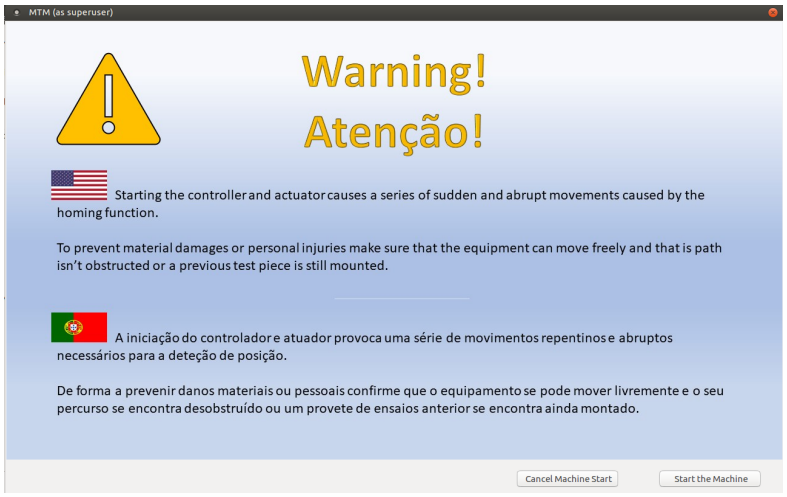

Figura 4.16: Janela inicial do programa para iniciação dos módulos e aviso ao utilizador

Tendo sido iniciada ou cancelada a alimentação dos módulos, a janela inicial irá encerrar e permitir ao utilizador acesso ao ambiente principal de programação de ensaios, sendo a inicialização possível a qualquer altura através do botão "*Start*". O botão "*Stop*", quando pressionado, desliga todos os módulos, funcionando como botão de emergência gráfico presente no ambiente de programação.

Estando todos os módulos iniciados, o utilizador fica possibilitado de gravar um novo teste ou mover o atuador. Os botões disponíveis para o efeito apenas permitem interação com o utilizador enquanto o controlador estiver a dar sinal de estar ligado. Esta verificação funciona através da requisição de envio e posteriormente da devida leitura de uma *Statusword*, considerando o controlador desligado sempre que uma resposta não é recebida a tempo. A montagem do provete é realizada recorrendo aos botões de movimento, sendo que estes permitem movimentos mais extensos para avanço rápido e movimentos mais lentos e de menor dimensão para maior precisão de montagem.

Para gravar um novo teste, como o que pode ser observado na Figura 4.17, é necessário introduzir o nome do utilizador, um nome para o ficheiro de texto que será gerado automaticamente, pelo menos uma condição de fim de teste e os parâmetros requeridos para o ensaio selecionado. Para maior facilidade de comparação entre as análises às duas soluções, programou-se neste exemplo um ensaio cíclico de atuação em força.

Durante a comunicação entre processos, enquanto o ficheiro de texto é gerado e os parâmetros gravados, os botões que funcionam por comunicação com o controlador ficam desativados. Após a conclusão destes processos, e enquanto as macros de teste são geradas e enviadas, é apresentada uma mensagem com a informação de sucesso da gravação dos parâmetros, como é observável na Figura 4.18. Simultaneamente com a apresentação da mensagem, as macros necessárias para a execução do teste são geradas

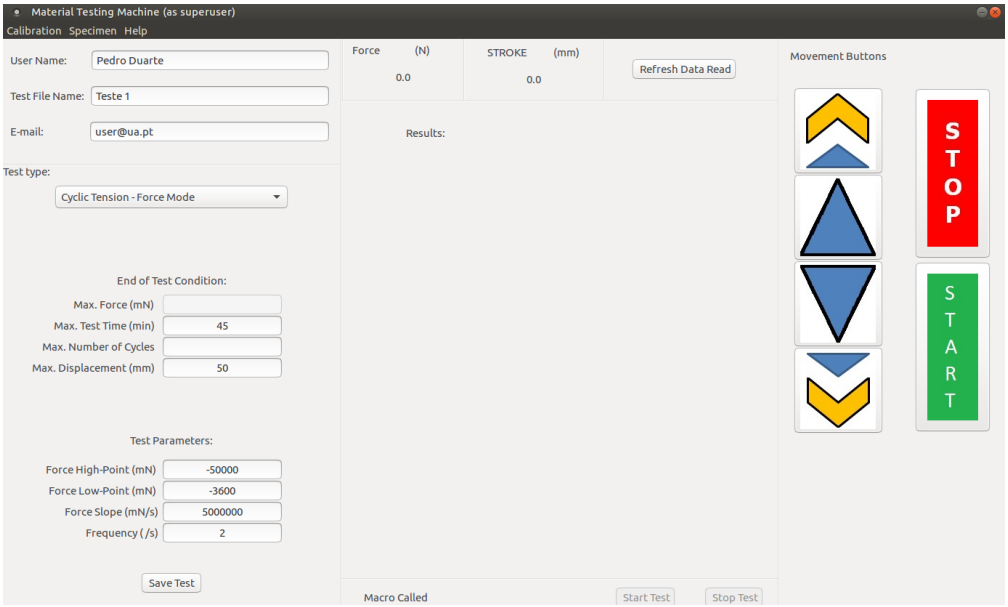

Figura 4.17: Janela principal de programação e requisição de testes, com os parâmetros de teste introduzidos

Figura 4.18: Mensagem com a informação de parâmetros guardados com sucesso

e enviadas para o controlador. O código de criação e envio destas macros encontra-se disponível no Anexo C.

Quando o envio das macros é finalizado, aparece na caixa de mensagens a informação de macros gravadas com sucesso, como se pode observar na Figura 4.19, e é reativada a utilização dos botões de movimento.

Para começar o teste, o utilizador apenas precisa de premir no botão "*Start Test*". O teste acaba quando umas das condições de fim é atingida, quando o botão de emergência é pressionado ou quando o botão "*Stop Test* é pressionado.

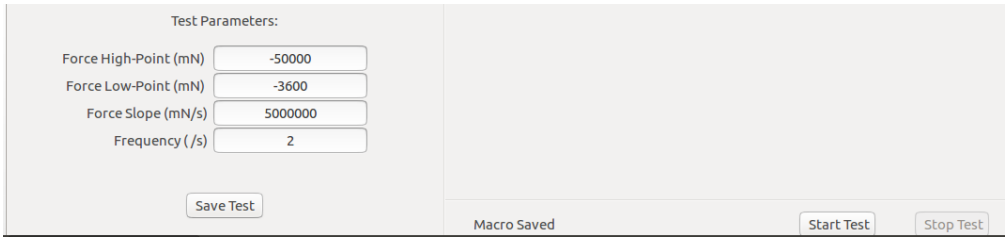

Figura 4.19: Mensagem com a informação de teste gravado com sucesso

Após o ensaio estar terminado, o ficheiro de texto com os resultados fica disponível ao utilizador no ambiente de trabalho do *Raspberry*, contendo toda a informação recolhida.

O resultado abaixo apresentado é representativo e resultado de um outro ensaio realizado na máquina, contendo as informações de teste.

" Test Log File:

Username: Pedro Duarte

Test Parameters:(Cyclic tension- Force mode)
-Force High = -36000 mN
-Force Low = -3600 mN
-Force Slope = 500000 mN/s
-Frequency = 4 times/s

End of test parameters:
-Maximum Cycles = 500 cycles

Load:-408.8mN Pos:69339nm
Load:3880.2mN Pos:69339nm
Load:21097.2mN Pos:69499nm
Load:19027.9mN Pos:69503nm
Load:10377.6mN Pos:69331nm
Load:12573.8mN Pos:69331nm
Load:21677.8mN Pos:69495nm
Load:17856.0mN Pos:69499nm

Load:13639.0mN Pos:69331nm
Load:8772.6mN Pos:69331nm
Load:22136.2mN Pos:69491nm
Load:21062.0mN Pos:69495nm
Load:8607.0mN Pos:69327nm"

Capítulo 5

Conclusões

5.1 Análise comparativa das soluções

Relembrando os objetivos mencionados no sub-capítulo 1.2 , este projeto teve como propósito melhorar o sistema existente de controlo e interação de uma máquina de ensaios baseada num atuador elétrico linear com encoder de alta resolução e que conta com um transdutor de força analógico.

Para executar essa melhoria pretendia-se utilizar um novo integrado de descodificação de encoder e um condicionador de transdutor de força, para além de uma interface Ethernet que pretendia facilitar a interação do utilizador com a máquina.

Através da análise da tabela 5.1, que contém uma descrição comparativa das soluções anterior e atual, é possível avaliar as diferenças entre os sistemas, facilitando assim a revisão de adequabilidade de resposta aos objetivos deste trabalho.

No módulo de leitura de dados, a solução implementada neste projeto apresenta maior fiabilidade e precisão na leitura através da utilização de novos equipamentos. Apresenta ainda uma solução mais compacta e intuitiva através do uso de uma placa de circuito impresso, oferecendo também uma maior proteção dos equipamentos através do uso de acopladores óticos, maior facilidade na deteção de danos e substituição de equipamentos em caso de ocorrência dos mesmos.

Outra grande diferença no módulo de leitura de dados é na calibração de célula de carga. A solução nova retira a necessidade de alteração da programação do microprocessador após cada calibração, possibilitando assim qualquer utilizador a efetuar esta operação através de comunicação Rs232.

No módulo de interface da máquina com o utilizador prescindiu-se da flexibilidade e controlo absoluto providenciado pelo software anteriormente utilizado por um sistema mais simples e fácil de utilizar, possibilitando ainda executar os ensaios mais comummente requeridos. A facilidade expressa advém do uso de um interface gráfico em vez do uso de um interface de programação.

O software criado simplifica também a recolha e gravação de dados e a comunicação com o respetivo módulo, automatizando e agrupando todos os processos num só software. A gravação dos requisitos do ensaio facilita também o pós tratamento dos dados e identificação dos testes executados.

Através das alterações implementadas no sistema da máquina de ensaios, conseguiu-se uma solução mais intuitiva, rápida, segura e fácil de utilizar que a solução anterior. Conseguiu-se também aplicar com sucesso o integrado de descodificação de encoder e um condicionador de transdutor de força numa placa desenvolvida para o propósito. Simplificou-se também o interface da máquina com o utilizador, não recorrendo para isso a uma ligação Ethernet como referido nos objetivos do trabalho por se ter encontrado uma solução mais prática e adequada às necessidades do utilizador durante o uso da mesma.

5.2 Propostas de desenvolvimento

Depois de analisada a solução desenvolvida que é tema deste trabalho, surgiram ideias de possíveis melhorias a realizar na solução implementada, de modo a providenciar um software mais completo e que permita tirar melhor partido de todas as possibilidades da máquina, aliando a simplicidade do sistema atual com a flexibilidade do sistema anterior.

É por isso proposto a continuação do desenvolvimento do interface gráfico neste trabalho apresentado, de modo a possibilitar ao utilizador a criação e desenvolvimento de testes por si desenvolvidos, que permitam realizar um ensaio material ao provete mais aproximado ás suas condições de funcionamento.

De modo a facilitar a utilização destes ensaios personalizados pelo utilizador, permitindo a sua mais fácil repetição, é proposto também a gravação e futura recuperação dos dados de teste antes programados.

É também proposto o desenvolvimento de um interface de calibração da célula de carga, que permita, através da atuação e monitorização de força do módulo de atuação, uma calibração fiável e certeira sem necessidade de meios externos à máquina.

Sugere-se ainda a execução automática de um relatório padrão após a finalização do ensaio, enviando o mesmo o através do email para o utilizador que requisitou o teste material, utilizando para isso a shield Ethernet mencionada no sub-capítulo 1.2 do trabalho.

Tabela 5.1: Tabela comparativa de soluções da máquina de ensaios

		Solução Prévia	Solução Desenvolvida
Equipamentos base	Atuação e controlo	Atuador e controlador SMAC	Atuador e controlador SMAC
	Obtenção de dados	Arduino Mega 2560, auxiliado do leitor de encoder HCTL-2032 para leitura de posição e recorrendo a uma porta analógica para leitura de carga	Arduino uno, auxiliado por uma placa de circuitos impressa que contém o leitor de encoder LS7366R para leitura da posição e o integrado HX711 para leitura de carga
	Interface	Software SMAC Control Center, em funcionamento num computador fixo	Software de interface gráfico dedicado a testes materiais, em funcionamento no Raspberry Pi
Obtenção de dados	Obtenção de leitura de posição	Comunicação paralela entre o Arduino Mega e o integrado HCTL-2032	Comunicação SPI entre o Arduino uno e o integrado LS7366R
	Obtenção de leitura de carga	Leitura através de uma porta analógica do Arduino Mega, após a filtragem do sinal	Comunicação I2C entre o integrado HX711 e o Arduino uno
	Calibração da célula de carga	Através de reprogramação do Arduino Mega, alterando a relação tensão elétrica medida/carga atuada, obtida através da reta de calibração	Através de comunicação Rs232 com o Arduino Uno
Interface	Flexibilidade do software	Software permite grande controlo sobre a atuação, permitindo o controlo de posição e força em cada instante de atuação	O software apenas permite realizar os testes de tração/compressão, flexão e fadiga
	Facilidade de ambientação e utilização	A criação e alteração de testes envolve conhecimento de programação no software	Definição de dados de teste simples e intuitiva
	Repetibilidade dos ensaios	Repetibilidade de ensaio com os mesmos parâmetros bastante simples, sendo apenas necessário correr a macro de início. A alteração de algum parâmetro de teste necessita de ambientação com o software	Para repetir um ensaio é necessário introduzir os parâmetros novamente, estando estes gravados no ficheiro de texto gerado pelo ensaio anterior
	Velocidade de programação de testes	Para a programação de um novo teste é necessário um conhecimento elevado de programação no software	Para a programação do teste basta introduzir os parâmetros requeridos
	Visualização e obtenção de dados	É necessário estabelecer comunicação por porta série do computador, os dados têm de ser copiados do histórico e gravado em ficheiro de texto	O software cria o ficheiro de texto e grava os dados automaticamente

Bibliografia

(1) Callister WD, Rethwisch DG. Materials Science and Engineering An Introduction. Sétima edi ed. United States of America: John Wiley & Sons, Inc.; 2007. Available from: https://abmpk.files.wordpress.com/2014/02/book{_}maretial-science-callister.pdf.

(2) Hughes A, Drury B. Electric Motors and Drives. 3rd ed. Oxford: Newnes; 2013. Available from: http://www.emic-bg.org/files/Electric{_}Motors{_}{_}{_}Drives.pdf.

(3) Yoichi M. Applications of Piezoelectric Actuator. Nec Technical Journal. 2005;1(Special Issue : Electronic Devices):82–86. Available from: http://www.nec.com/en/global/techrep/journal/g06/n05/pdf/t060519.pdf.

(4) Fischer-Cripps T. Newnes Interfacing Companion: Computers, Transducers, Instrumentation and Signal Processing. Oxford: A.C. Fischer-Cripps; 2002. Available from: https://books.google.pt/books/about/Newnes{_}Interfacing{_}Companion.html?id=Xms9p5xmbVwC{&}source=kp{_}cover{&}redir{_}esc=y.

(5) Rizzoni G. Principles and Applications of Electrical Engineering. Boston: McGraw-Hill Higher Education; 2000.

(6) Capacitive transducers;. Available from: http://www.idc-online.com/technical{_}references/pdfs/electronic{_}engineering/Capacitive{_}Transducers.pdf.

(7) Beach JN, Gebelein RE, Longren S, Mantua L, McMillan GK, Morrow TA, et al. Geometric And Motion Sensors. In: McMillan GK, Considine DM, editors. Process/Industrial instruments and controls Handbook. 5th ed. New York, USA: McGraw-Hill, L.da; 1999. p. 119.

(8) Fraden J. Handbook of modern sensors: Physics, Designs, and Application. 3rd ed. Cambridge: Springer-Verlag, Inc.; 2004.

(9) inelta Sensorsysteme GmbH & Co. Serie KMM65,Kraftsensoren / Force Sensors: DATASHEET. Munique, Alemanha: inelta Sensorsysteme GmbH & Co;. Available from: http://www.inelta.de/fileadmin/user{_}upload/daten/Datenblaetter/Kraft/inelta{_}datenblatt{_}kraft-kmm65{_}3199.pdfwww.inelta.de.

(10) SMAC. Control Tuning: TECHNICAL NOTE. California: SMAC Corporation; 2017. Available from: http://www.smac-mca.com/documents/controllers/TechNote-Tuning-LCC.pdf.

(11) SMAC; INGENIA-CAT. Embedded Motion Control Library: TECHNICAL NOTE. California: SMAC Corporation; 2017. Available from: www.smac-mca.comwww.ingeniamc.com.

(12) SMAC. SMAC Control Center Training v1.2: TECHNICAL NOTE. California: SMAC Corporation; 2017. Available from: http://www.smac-mca.com/documents/PDFs/SMACControlCenterTraining{_}v1.2.pdf.

(13) Arduino. No Title;. Available from: https://www.arduino.cc/en/Products/Compare.

(14) Raspberry Pi 3 Model B - Raspberry Pi;. Available from: https://www.raspberrypi.org/products/raspberry-pi-3-model-b/.

(15) Askeland DR, Fulay PP, Wright WJ. The Science And Engineering of Materials. 6th ed. Stamford: Cengage Learning; 2010. Available from: http://home.ufam.edu.br/berti/nanomateriais/0495296023MaterialsEnginee.pdf.

(16) Smith WF. Príncipios De Ciência e Engenharia Dos Materiais. Terceira e ed. Amadora: McGraw-Hill, L.da; 1998.

(17) ASM Handbook: Volume 8, Mechanical Testing. 4th ed. Ohio: American Society for Metals; 1992.

(18) Cruz A, Carreira J. Ensaios Mecânicos. 1st ed. Lisboa: Instituto de Soldadura e Qualidade; 1992.

(19) Parr A. Hydraulics and Pneumatics: A Technician's and engineer's guide. Oxford: Newnes; 1997.

(20) Sclater N, Chironis NP. Mechanisms and Mechanical devices sourcebook. 4th ed. New York: McGraw-Hill; 2007. Available from: http://160592857366.free.fr/joe/ebooks/MechanicalEngineeringBooksCollection/THEORYOFMACHINES/MECHANISMSANDMECHANICALDEVICES4e.pdf.

(21) Fialho AB. Automação Pneumática: Projetos, Dimensionamento e Análise de Circuitos. 6th ed. São Paulo: Editora Érica Ltda.; 2008.

(22) Fialho AB. Automação hidráulica: projetos, dimensionamento e análise de circuitos. 5th ed. São Paulo: Editora Érica Ltda.; 2007.

(23) Callear BJ, Pinches MJ. Power pneumatics. Hertfordshire: Prentice Hall Europe; 1997.

(24) Bao MH. Micro Mechanical Transducers: Pressure Sensors, Accelerometers and Gyroscopes. In: Middelhoek S, editor. Handbook of Sensors and Actuators. 1st ed. Amsterdão: Elsevier Science B.V.; 2000. p. 378.

(25) SMAC. Measuring forces with LCC: TECHNICAL NOTE. California: SMAC Corporation; 2017. Available from: http://www.smac-mca.com/documents/controllers/TechNote-Forces-LCC.pdf.

(26) Atmel Corporation. ATmega640/V-1280/V-1281/V-2560/V-2561/V: DATASHEET. San Jose, California: Atmel Corporation; 2014. Available from: http://www.atmel.com/Images/Atmel-2549-8-bit-AVR-Microcontroller-ATmega640-1280-1281-2560-2561{_}datasheet.pdfwww.atmel.com.

(27) Arduino. Arduino MEGA 2560: SCHEMATIC DRAWING. Arduino;. Available from: https://www.arduino.cc/en/uploads/Main/arduino-mega2560{_}R3-sch.pdf.

(28) G Ring. LCC-10 Outline Drawing. California: SMAC Corporation; 2017. Available from: http://www.smac-mca.com/documents/PDFs/LCC-10.pdf.

(29) AVAGO. HCTL-2032, HCTL-2032-SC, HCTL-2032-SCT, HCTL-2022: DATA SHEET. AVAGO TECHNOLOGIES; 2007. Available from: www.avagotech.com.

(30) Åström, Karl Johan; Hägglund T. PID controllers: theory, design, and tuning. 2nd ed. North Carolina: ISA - The Instrumentation, Systems and Automation Society; 2006.

(31) SMAC. SMAC Control Center Get Started Manual: TECHNICAL NOTE. California: SMAC Corporation; 2017. Available from: http://www.smac-mca.com/documents/PDFs/SMACControlCenterGettingStartedv1.2.pdf.

(32) Gaggero AA. Csvconvert: A simple command to gather comma-separated value files into Stata. Stata Journal. 2014;14(3):662–669. Available from: https://www.scopus.com/record/display.uri?eid=2-s2.0-85001020580{&}origin=resultslist{&}sort=plf-f{&}src=s{&}st1=.csv+AND+files+{&}st2={&}sid=5DE8D3A91F0EDF3E7436085E5C15C41C.wsnAw8kcdt7IPYLO.

(33) SMAC; INGENIA-CAT. MotionLab Software: SOFTWARE MANUAL. California: INGENIA-CAT; 2012. Available from: www.ingeniamc.comhttp://www.smac-mca.com/documents/PDFs/MotionLab.pdf.

(34) AVIA. HX711, 24-Bit Analog-to-Digital Converter (ADC) for Weigh Scales: DATASHEET. Xiamen, China: Avia Semiconductor(Xiamen)Co.,Ltd.;. Available from: https://www.sparkfun.com/products/13879.

(35) LSI Computer Systems I. LS7366R,32-BIT QUADRATURE COUNTER WITH SERIAL INTERFACE : DATASHEET. New York, USA: LSI Computer Systems, Inc; 2014. Available from: http://www.lsicsi.com/pdfs/Data{_}Sheets/LS7366R.pdf.

(36) Vishay. High Speed Optocoupler, 1 MBd, Photodiode with Transistor Output. PensilvÂnia: Vishay; 2015. Available from: https://www.vishay.com/docs/83604/6n135.pdf.

Apêndice A

Código do módulo de leitura de dados

```
/* Arduino Uno
Created by: Pedro Cruz Duarte
Arduino module to handle the reading and register of the load cell and position decoder
HX711 - Load cell reader
LS7366R - Encoder reader
*/

#include <HX711.h>
#include <SPI.h>
#include <EEPROM.h>

// LS7366 operation code list
#define CLR_MDR0 0x08    // Clear mode register 0
#define CLR_MDR1 0x10    // Clear mode register 1
#define CLR_CNTR 0x20    // Clear counter
#define CLR_STR 0x30     // Clear status register
#define RD_MDR0 0x48     // Read mode register 0
#define RD_MDR1 0x50     // Read mode register 1
#define RD_CNTR 0x60     // Read counter
#define RD_OTR 0x68      // Read output register
#define RD_STR 0x70      // Read status register
#define WR_MDR1 0x90     // Write mode register 1
#define WR_MDR0 0x88     // Write mode register 0
#define WR_DTR 0x98      // Write data register/comparator
#define LOAD_CNTR 0xE0   // Load counter
#define LOAD_OTR 0xE4    // Load output register

// LS7366R auxiliary pins and variables
#define SS 10            // Slave select is set on Pin 10
#define CNT_EN 4         // Count enable is set on Pin 4
byte MDR0;               // Mode register 0: sets the operating mode for the LS7366R
byte MDR1;               // Mode register 1: regiter that appends to the MDR0, it mainly handles flags
byte nothing = 0x00;     // Random data to receive response in the SPI COM
long data=0;             // Position received from the encoder

// HX711 auxiliary pins and variables
```

```
37 #define DOUT 3          // Data from the load cell is received in Pin 3
38 #define CLK 2           // Clock pin for communication is Pin 2
39 HX711 scale(DOUT, CLK); // Definition of the pins needed for the
       communication
40 float calibration_factor;// Calibration factor for the load cell reader
41 long zero_factor;       // Zeroes the load cell read
42 uint8_t *address = 0;   // Adress to hold the calibration factor in the
       EEPROM
43
44 // LCC-11 auxiliary pins and variables
45 #define Freq_Pin 8      // Connection with the controller to allow the
       calculation of the frequency
46 #define End_Pin 9       // End of test selection pin
47
48 // Auxiliary variables to the communication
49 unsigned char Data[4];
50
51
52 void setup()    {
53     // Serial monitor configuration
54     Serial.begin(9600);     // Begin communication with a serial port by
       Rs232 with 9600bps baudrate
55
56     // EEprom data retrieval
57     if (eeprom_read_byte(address + 1)) // If the next byte to the
       calibration factor is 1 it means that the value is negative
58     { calibration_factor = -1 * eeprom_read_byte(address);}
59     else {  calibration_factor = eeprom_read_byte(address); }
60
61     // Inicialization of the load cell
62     scale.set_scale();                  // Set the scale to a readable value
63     scale.tare();                       // Reset the scale to 0
64     zero_factor=scale.read_average(); // Get a baseline reading
65     scale.set_scale(calibration_factor); // Adjusts the scale factor to the
        calibration factor
66
67     // SPI communication configuration
68     SPI.begin();                        // Inicialize SPI communication
69     SPI.setDataMode(SPI_MODE0);         // Sets the clock polarity and phase
       and configures the output edge and data capture timings
70     SPI.setBitOrder(MSBFIRST);          // Sets the order of the bits shifted
        out of and into the SPI bus (Most-significant bit first)
71     SPI.setClockDivider(SPI_CLOCK_DIV8); // Sets the SPI clock divider
       relative to the system clock
72
73     pinMode(CNT_EN, OUTPUT);            // Count enable as output
74     digitalWrite(CNT_EN, HIGH);         // Enable Counting
75     pinMode(SS, OUTPUT);                // Slave Select as output
76     digitalWrite(SS, HIGH);             // Slave select as active
77     delay(100);
78     SPI_Configuration();                // Configuration of the mode register
        0 and 1 bytes of the quadrature counter with serial interface
79
80     // SMAC Controller initiation
81     pinMode(Freq_Pin, INPUT);           // Sets the frequency pin as input
82     pinMode(End_Pin, INPUT);            // Set the calibration selection as
       input
83
```

```
84 }
85
86
87 long Read_Position(void) { // Receive the position from the decoder LS7366R
88
89     long position;
90     unsigned int position1;
91     unsigned int position0;
92     unsigned int position2;
93     unsigned int position3;
94
95     digitalWrite(SS, LOW);              // Initiate slave communication
96     SPI.transfer(RD_CNTR);              // Transfer CNTR to OTR, then output
       OTR serially on TXD (MISO)
97     position3 = SPI.transfer(nothing);//
98     position2 = SPI.transfer(nothing);//
99     position1= SPI.transfer(nothing); //
100    position0=SPI.transfer(nothing);  //
101    digitalWrite(SS, HIGH);             // End slave communication
102
103    position= ((long)position3 << 24) + ((long)position2 << 16) + ((long)
       position1 << 8) + (long)position0;
104    return position;
105 }//Read_Encoder
106
107
108 void SPI_Configuration(void)    {
109     /**
110         SPI Communication: Configuration of the LS7366R quadrature decoder
111         Configuration of the mode register 0 and 1 bytes of the quadrature
       counter with serial interface
112     */
113
114     digitalWrite(SS, LOW); // Initiate slave communication
115     SPI.transfer(WR_MDR0); // Write into Mode Register 0
116     SPI.transfer(0x61);    // MDR0=01100001=0x61
117                            //      ------01: x1 quadrature count mode
118                            //      ----00--: free-running count mode
119                            //      --10----: configure index (Z+) as the
       reset CNTR input (clears CNTR to 0)
120                            //      -1------: synchronous index
121                            //      0-------: filter clock division factor=1
122     digitalWrite(SS, HIGH);// End slave communication
123
124     digitalWrite(SS, LOW);
125     SPI.transfer(RD_MDR0); // Read Mode Register 0
126     MDR0 = SPI.transfer(nothing); // Data received
127     digitalWrite(SS, HIGH);
128
129     digitalWrite(SS, LOW); // Initiate slave communication
130     SPI.transfer(WR_MDR1); // Write into Mode Register 1
131     SPI.transfer(0x00);    // MDR1=00000000=0x00
132                            //      ------00: 4-byte counter mode
133                            //      -----0--: enable counting
134                            //      ----0---: not used
135                            //      0000----: Flags (CY, BW, CMP, IDX)
136     digitalWrite(SS, HIGH); // End slave communication
137
```

```
138     digitalWrite(SS, LOW);
139     SPI.transfer(RD_MDR1); // Read Mode Register 1
140     MDR1 = SPI.transfer(nothing); // Data received
141     digitalWrite(SS, HIGH);
142 }// SPI_Configuration
143 
144 
145 void SendData(void)     {
146     Serial.write("Load:");
147     Serial.print(scale.get_units()*-0001, 1); // Prints the force applied
    in the load cell
148     Serial.write("mN ");
149     data = Read_Position() / -0.24944;        // Position reading and
    decoding to human readable number
150     Serial.write("  Pos:");
151     Serial.print(data);
152     Serial.write("mm");
153     Serial.println();
154     delay(50);
155 }// SendData
156 
157 
158 void loop(){
159     if (digitalRead(Freq_Pin) == HIGH) { SendData();}
160     if (digitalRead(End_Pin) == HIGH) { Serial.println("TestEnded"); }
161 }// loop
162 
163 
164 void serialEvent()  {          // Interrupt upon receiving data
165     if (Serial.available()) { // Function to analyse if it as received any
    data in via serial port and process it
166 
167         char temp = Serial.read();
168         if (temp == '+')    {
169             calibration_factor += 1;  // Adds to a calibration factor
170             scale.set_scale(calibration_factor); // Adjusts the scale
    factor to the calibration factor
171         }
172 
173         else if (temp == '-')   {
174         calibration_factor -= 1; // Subtracts from a calibration factor
175         scale.set_scale(calibration_factor); // Adjusts the scale factor to
     the calibration factor
176         }
177 
178         else if (temp == 's')   {
179             SendData();
180         }
181 
182         else if (temp == 'c')   {
183             Serial.print("Zero Factor:");
184             Serial.print(zero_factor);
185             Serial.print(" CalibrationFactor:");
186             Serial.println(calibration_factor);
187 
188             // Save calibration factor into the EEPROM
189             if (calibration_factor > 0) {
190                 eeprom_write_byte(address, byte(calibration_factor));
```

```
191                 eeprom_write_byte(address + 1, 0);
192             }   else {
193                 eeprom_write_byte(address, byte(-1*calibration_factor));
194                 eeprom_write_byte(address + 1, 1);
195             }
196         }
197     }//if Serial is available
198 }
```

Apêndice B

Código de início da aplicação de interface com o utilizador

```
/**
 *  @file Main.c
 *  @brief Ficheiro com a base do programa
 *
 *  Ficheiro com a estrutura base do código que cria os dois processos necessários ao funcionamento da máquina: Um processo de comunicação com hardware externo e gestão de funcionamento e um de interface gráfica com o utilizador
 *
 *  @author Pedro Duarte, duarte.pedro@ua.pt
 *
 *  @internal
 *      Created 2017
 *      Company Universidade de Aveiro (University of Aveiro)
 *      Copyright Copyright(c) 2017, Pedro Duarte
 *
 *  ================================================================================
 */


// Header file that contains the include of every other library
#include "library.h"
// Automatically created header file that contains the list of functions created in the project files
#include "prototypes.h"

int shr_mem_id;
int child_pid;
int parent_pid;


int main (int argc, char *argv[]){
    int pid=fork();                                                 // Forks the program in two processes: A parent and a child; who run independent from wich other but can communicate
    /* Process fork:
     * Parent- Handles the interface of the program
```

```
32      * Child- Handles the communications and the piGPIO
33      */
34
35     if(pid == -1) {      // Failed to fork
36         printf("Couldn't fork(). Exiting\n");   // Print Error in the
    terminal
37         return -1;
38     }
39
40     if (pid == 0) {                      // Child of the fork function
41     // ******************** Initialize Libraries ***********************
42         wiringPiSetup();
43         gpioTerminate();
44         if(gpioInitialise()<0) { printf("Fail to initialise piGPIO\n");
    return -1; }
45     // ***************** Initialize Communications *********************
46         Start_UART1();                   // Start the UART1
47         Start_UART2();                   // Start the UART2
48     // ************************ Link Signals ****************************
49         signal(SIGUSR1,Child_Kill);      // Links the SIGUSR1 signal to the
    Child_Kill function on TestMachine.c
50         signal(SIGUSR2,ChildReadMem);    // Links the SIGUSR2 signal to the
    ChildReadMem function
51         parent_pid=getppid();            // Get the pid from the parent to
    allow to send the SIGUSR2 signal
52         shr_mem_id=SharedMem_Get();      // Get the shared memory id
53         ChildFunction();                 // Calls the main function from the
     TestMachine.c
54     }
55
56     else {                               // Parent of the fork function
57         signal(SIGINT,Intercept_kill);   // Intercepts the kill signal and
    runs the function
58         signal(SIGUSR2,ParentReadMem);   // Links the SIGUSR2 signal to the
    ParentReadMem function
59         shr_mem_id=SharedMem_Get();      // Get the shared memory id
60         child_pid=pid;                   // Set the child pid to allow to
    send the SIGUSR2 signal
61         ParentFunction(argc,argv);       // Calls the Parent function
62     }
63     return 0;
64 }//main
```

Apêndice C

Código para a geração e envio das macros específicas a um ensaio cíclico por atuação em força

```
case 20: // Ensaio cíclico - força
// Macro 20
Send_UART2(End_Program);          // End program
Send_UART2("0x20 W 0x042C04 20\r");       // Reset macro 20
delay(600);
Send_UART2("0x20 W 0x012C05 20\r");       // [MacroNumber 20] Ensaios
    Cíclicos
Send_UART2("0x20 W 0x022C05 0\r");        // Command 0
Send_UART2("0x20 W 0x032C05 0x162C014000006064\r");  // [SetVariable Var=
    General_purpouse_registers-W30(0x1E2C00),Subtract,Variable_and_constant
    ,Var1=Position_actual_value(0x006064),Const=20000]{Guardar no apontador
     W30 a posição final}Get Position_actual_value(0x006064) in ACCUM
Send_UART2("0x20 W 0x022C05 1\r");        // Command 1
snprintf(str5,sizeof str5 ,"0x20 W 0x032C05 0x072C0140%08X\r",MaxStroke);
    // Subtract 20000 from ACCUM
Send_UART2(str5);
Send_UART2("0x20 W 0x022C05 2\r");        // Command 2
Send_UART2("0x20 W 0x032C05 0x152C0140001E2C00\r");  // Write ACCUM to
    General_purpouse_registers-W30(0x1E2C00)
Send_UART2("0x20 W 0x022C05 3\r");        // Command 3
Send_UART2("0x20 W 0x032C05 0x0D2C02400000004B\r");  // [Wait Time,Timeout
    =75]{Atraso para certeza da gravação da variável}
Send_UART2("0x20 W 0x022C05 4\r");        // Command 4
Send_UART2("0x20 W 0x032C05 0x282C004000000000\r");  // [SetVariable Var=
    General_purpouse_registers-W40(0x282C00),Constant,Const=0]{Reset do
    contador pequeno}
Send_UART2("0x20 W 0x022C05 5\r");        // Command 5
Send_UART2("0x20 W 0x032C05 0x0D2C02400000004B\r");  // [Wait Time,Timeout
    =75]{Atraso para certeza da gravação da variável}
Send_UART2("0x20 W 0x022C05 6\r");        // Command 6
Send_UART2("0x20 W 0x032C05 0x292C004000000000\r");  // [SetVariable Var=
    General_purpouse_registers-W41(0x292C00),Constant,Const=0]{Reset do
    contador Grande}
Send_UART2("0x20 W 0x022C05 7\r");        // Command 7
Send_UART2("0x20 W 0x032C05 0x0D2C02400000004B\r");  // [Wait Time,Timeout
    =75]{Atraso para certeza da gravação da variável}
```

```
Send_UART2("0x20 W 0x022C05 8\r");       // Command 8
Send_UART2("0x20 W 0x032C05 0x032C044000000015\r");  // Sequence Macro Jump,MacroNumber=21


//__________Macro 21___________
Send_UART2(End_Program);          // End program
Send_UART2("0x20 W 0x042C04 21\r");       // Reset macro 21
delay(600);
Send_UART2("0x20 W 0x012C05 21\r");       // [MacroNumber 21]{Rampa de subida}
Send_UART2("0x20 W 0x022C05 0\r");       // Command 0
snprintf(str5,sizeof str5 ,"0x20 W 0x032C05 0x00608740%08X\r",ForceSlope/36); // ForceSlope // [ForceMove Target=-1000,Slope=50000]{Aplica força de tração ao provete}
Send_UART2(str5);
Send_UART2("0x20 W 0x022C05 1\r");       // Command 1
Send_UART2("0x20 W 0x032C05 0x012C004000000009\r");  // Set profile force mode first. Prepare ACCUM to call system macro #9 : FORCE_MOVE
Send_UART2("0x20 W 0x022C05 2\r");       // Command 2
Send_UART2("0x20 W 0x032C05 0x012C044000000003F\r");  // Call system macro
Send_UART2("0x20 W 0x022C05 3\r");       // Command 3
char str4[10];
snprintf(str4,sizeof str4 ,"%4X",ForceHigh/36);    // ForceHigh // Apply target force
str4[0]=str4[1]=str4[2]=str4[3]=48;
snprintf(str5,sizeof str5 ,"0x20 W 0x032C05 0x00607120%s\r",str4); // ForceHigh // Apply target force
Send_UART2(str5);
Send_UART2("0x20 W 0x022C05 4\r");       // Command 4
snprintf(str5,sizeof str5 ,"0x20 W 0x032C05 0x0D2C0240%08X\r",500/Frequency); // Frequency  // [Wait Time,Timeout=20]
Send_UART2(str5);
Send_UART2("0x20 W 0x022C05 5\r");       // Command 5
Send_UART2("0x20 W 0x032C05 0x172C014000022A02\r");  // [SetOutput Digital_output_0,On]{Ativa saída para cálculo da frequência}Get value of digital output in ACCUM
Send_UART2("0x20 W 0x022C05 6\r");       // Command 6
Send_UART2("0x20 W 0x032C05 0x052C01400000FFFE\r");  // Set output bit to 0
Send_UART2("0x20 W 0x022C05 7\r");       // Command 7
Send_UART2("0x20 W 0x032C05 0x152C014000022A02\r");  // Write ACCUM back to digital outputs
Send_UART2("0x20 W 0x022C05 8\r");       // Command 8
Send_UART2("0x20 W 0x032C05 0x0D2C024000000064\r");  // [Wait Time,Timeout=100]{Atraso para permitir ao Arduino apanhar a mudança de estado, dá a frequência}
Send_UART2("0x20 W 0x022C05 9\r");       // Command 9
Send_UART2("0x20 W 0x032C05 0x0060644100000000\r");  // [GetVariable Var=Position_actual_value(0x006064)]
Send_UART2("0x20 W 0x022C05 10\r");       // Command 10
Send_UART2("0x20 W 0x032C05 0x172C014000022A02\r");  // [SetOutput Digital_output_0,Off]{Desativar a saída digital}Get value of digital output in ACCUM
Send_UART2("0x20 W 0x022C05 11\r");       // Command 11
Send_UART2("0x20 W 0x032C05 0x062C014000000001\r");  // Set output bit to 1
Send_UART2("0x20 W 0x022C05 12\r");       // Command 12
Send_UART2("0x20 W 0x032C05 0x152C014000022A02\r");  // Write ACCUM back to digital outputs
```

```
65 Send_UART2("0x20 W 0x022C05 13\r");       // Command 13
66 Send_UART2("0x20 W 0x032C05 0x032C044000000016\r");  // Sequence Macro Jump
       ,MacroNumber=22
67
68
69 //__________Macro 22_____________
70 Send_UART2(End_Program);          // End program
71 Send_UART2("0x20 W 0x042C04 22\r");       // Reset macro 22
72 delay(600);
73 Send_UART2("0x20 W 0x012C05 22\r");        // [MacroNumber 22]{Rampa de
       descida}
74 Send_UART2("0x20 W 0x022C05 0\r");
75 snprintf(str5,sizeof str5 ,"0x20 W 0x032C05 0x00608740%08X\r",ForceSlope
       /36); // ForceSlope // [ForceMove Target=-100,Slope=50000]{Aplica força
        de tração ao provete}
76 Send_UART2(str5);
77 Send_UART2("0x20 W 0x022C05 1\r");
78 Send_UART2("0x20 W 0x032C05 0x012C004000000009\r");  // Set profile force
       mode first. Prepare ACCUM to call system macro #9 : FORCE_MOVE
79 Send_UART2("0x20 W 0x022C05 2\r");
80 Send_UART2("0x20 W 0x032C05 0x012C044000000003F\r");  // Call system macro
81 Send_UART2("0x20 W 0x022C05 3\r");
82 snprintf(str4,sizeof str4 ,"%4X",ForceLow/36);    // ForceHigh // Apply
       target force
83 str4[0]=str4[1]=str4[2]=str4[3]=48;
84 snprintf(str5,sizeof str5 ,"0x20 W 0x032C05 0x00607120%s\r",str4); //
       ForceHigh // Apply target force
85 Send_UART2(str5);
86 Send_UART2("0x20 W 0x022C05 4\r");
87 snprintf(str5,sizeof str5 ,"0x20 W 0x032C05 0x0D2C0240%08X\r",500/Frequency
       ); // Frequency  // [Wait Time,Timeout=20]
88 Send_UART2(str5);
89 Send_UART2("0x20 W 0x022C05 5\r");
90 Send_UART2("0x20 W 0x032C05 0x172C014000022A02\r");  // [SetOutput
       Digital_output_0,On]Get value of digital output in ACCUM
91 Send_UART2("0x20 W 0x022C05 6\r");
92 Send_UART2("0x20 W 0x032C05 0x052C01400000FFFE\r");  // Set output bit to 0
93 Send_UART2("0x20 W 0x022C05 7\r");
94 Send_UART2("0x20 W 0x032C05 0x152C014000022A02\r");  // Write ACCUM back to
        digital outputs
95 Send_UART2("0x20 W 0x022C05 8\r");
96 Send_UART2("0x20 W 0x032C05 0x0D2C024000000064\r");  // [Wait Time,Timeout
       =100]
97 Send_UART2("0x20 W 0x022C05 9\r");
98 Send_UART2("0x20 W 0x032C05 0x0060644100000000\r");  // [GetVariable Var=
       Position_actual_value(0x006064)]
99 Send_UART2("0x20 W 0x022C05 10\r");
100 Send_UART2("0x20 W 0x032C05 0x172C014000022A02\r");  // [SetOutput
       Digital_output_0,Off]Get value of digital output in ACCUM
101 Send_UART2("0x20 W 0x022C05 11\r");
102 Send_UART2("0x20 W 0x032C05 0x062C014000000001\r");  // Set output bit to 1
103 Send_UART2("0x20 W 0x022C05 12\r");
104 Send_UART2("0x20 W 0x032C05 0x152C014000022A02\r");  // Write ACCUM back to
        digital outputs
105 Send_UART2("0x20 W 0x022C05 13\r");
106 Send_UART2("0x20 W 0x032C05 0x032C044000000017\r");  // Sequence Macro Jump
       ,MacroNumber=23
107
```

```
108
109 //__________Macro 23___________
110 Send_UART2(End_Program);          // End program
111 Send_UART2("0x20 W 0x042C04 23\r");       // Reset macro 23
112 delay(600);
113 Send_UART2("0x20 W 0x012C05 23\r");       // [MacroNumber 23]{Parte 1-
    Avaliação de fim de teste por número de ciclos}
114 Send_UART2("0x20 W 0x022C05 0\r");
115 Send_UART2("0x20 W 0x032C05 0x162C014000006064\r");  // [If Variable,Var=
    Position_actual_value(0x006064),Lower,Var2=General_purpouse_registers-
    W30(0x1E2C00),Then_Macro_Call=50,Else_Macro_Continue]{Avalia fim de
    teste por deformação}Get variable in ACCUM
116 Send_UART2("0x20 W 0x022C05 1\r");
117 Send_UART2("0x20 W 0x032C05 0x132C0240001E2C00\r");  // Do comparison with
    variable
118 Send_UART2("0x20 W 0x022C05 2\r");
119 Send_UART2("0x20 W 0x032C05 0x012C044000000032\r");  // Then call macro 50
120 Send_UART2("0x20 W 0x022C05 3\r");
121 Send_UART2("0x20 W 0x032C05 0x162C014000282C00\r");  // [If Variable,Var=
    General_purpouse_registers-W40(0x282C00),Higher,Const=499,
    Then_Macro_Call=24,Else_Macro_Continue]{CondiÃ§Ã£o de fim de teste por
    mÃ¡ximo de ciclos.1}Get variable in ACCUM
122 Send_UART2("0x20 W 0x022C05 4\r");
123 Send_UART2("0x20 W 0x032C05 0x052C0240000001F3\r");  // Do comparison with
    constant
124 Send_UART2("0x20 W 0x022C05 5\r");
125 Send_UART2("0x20 W 0x032C05 0x012C044000000018\r");  // Then call macro 24
126 Send_UART2("0x20 W 0x022C05 6\r");
127 Send_UART2("0x20 W 0x032C05 0x162C014000282C00\r");  // [SetVariable Var=
    General_purpouse_registers-W40(0x282C00),Add,Variable_and_constant,Var1
    =General_purpouse_registers-W40(0x282C00),Const=1]{Incrementa um ciclo
    à variável}Get General_purpouse_registers-W40(0x282C00) in ACCUM
128 Send_UART2("0x20 W 0x022C05 7\r");
129 Send_UART2("0x20 W 0x032C05 0x012C014000000001\r");  // Add 1 to ACCUM
130 Send_UART2("0x20 W 0x022C05 8\r");
131 Send_UART2("0x20 W 0x032C05 0x152C014000282C00\r");  // Write ACCUM to
    General_purpouse_registers-W40(0x282C00)
132 Send_UART2("0x20 W 0x022C05 9\r");
133 Send_UART2("0x20 W 0x032C05 0x032C044000000015\r");  // Sequence Macro Jump
    ,MacroNumber=21 Volta ao inicio do ciclo
134
135
136 // __________Macro 24___________
137 Send_UART2(End_Program);          // End program
138 Send_UART2("0x20 W 0x042C04 24\r");       // Reset macro 24
139 delay(600);
140 Send_UART2("0x20 W 0x012C05 24\r");       // [MacroNumber 24]{Parte 2-
    Avaliação de fim de teste por número de ciclos}
141 Send_UART2("0x20 W 0x022C05 0\r");
142 Send_UART2("0x20 W 0x032C05 0x162C014000292C00\r");  // [If Variable,Var=
    General_purpouse_registers-W41(0x292C00),Higher,Const=999,
    Then_Macro_Call=50,Else_Macro_Continue]{Condição de fim de teste por
    máximo de ciclos.2}Get variable in ACCUM
143 Send_UART2("0x20 W 0x022C05 1\r");
144 snprintf(str5,sizeof str5,"0x20 W 0x032C05 0x052C0240%08X\r",MaxCycles
    /499); // Do comparison with constant
145 Send_UART2(str5);
```

```
146 Send_UART2("0x20 W 0x022C05 2\r");
147 Send_UART2("0x20 W 0x032C05 0x012C044000000032\r");  // Then call macro 50
148 Send_UART2("0x20 W 0x022C05 3\r");
149 Send_UART2("0x20 W 0x032C05 0x162C014000292C00\r");  // [SetVariable Var=
    General_purpouse_registers-W41(0x292C00),Add,Variable_and_constant,Var1
    =General_purpouse_registers-W41(0x292C00),Const=1]{Incrementa um ciclo
    à variável}Get General_purpouse_registers-W41(0x292C00) in ACCUM
150 Send_UART2("0x20 W 0x022C05 4\r");
151 Send_UART2("0x20 W 0x032C05 0x012C014000000001\r");  // Add 1 to ACCUM
152 Send_UART2("0x20 W 0x022C05 5\r");
153 Send_UART2("0x20 W 0x032C05 0x152C014000292C00\r");  // Write ACCUM to
    General_purpouse_registers-W41(0x292C00)
154 Send_UART2("0x20 W 0x022C05 6\r");
155 Send_UART2("0x20 W 0x032C05 0x282C004000000000\r");  // [SetVariable Var=
    General_purpouse_registers-W40(0x282C00),Constant,Const=0]{Reset do
    contador pequeno}
156 Send_UART2("0x20 W 0x022C05 7\r");
157 Send_UART2("0x20 W 0x032C05 0x032C044000000015\r");// Sequence Macro Jump,
    MacroNumber=21 Volta ao incio do ciclo
158 Send_UART2(End_Program);         // End program
159 return 1;
160 break;
```

Apêndice D

Esquema da placa de circuitos impressos desenvolvida para o módulo de leitura de dados

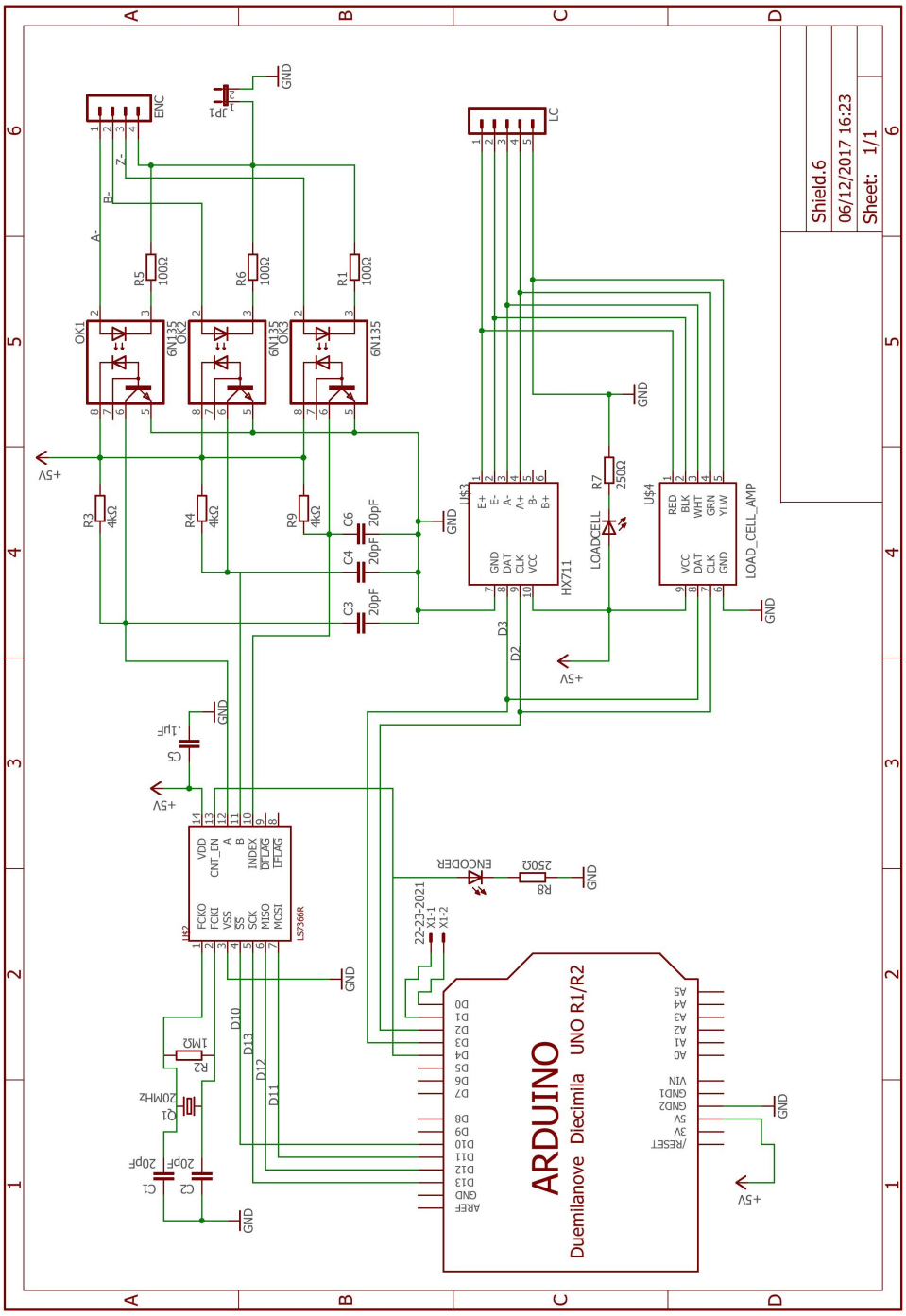

Figura D.1: Esquema elétrico da *shield* desenvolvida para a leitura de dados de uma célula de carga e encoder incremental

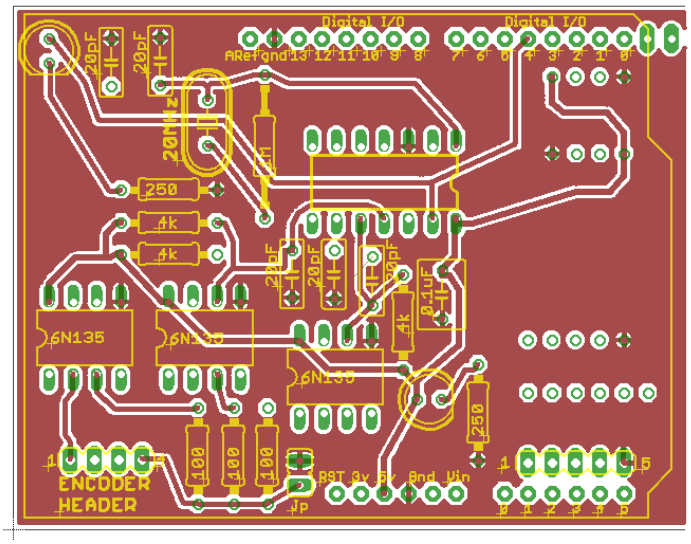

Figura D.2: Vista de topo da *shield* desenvolvida

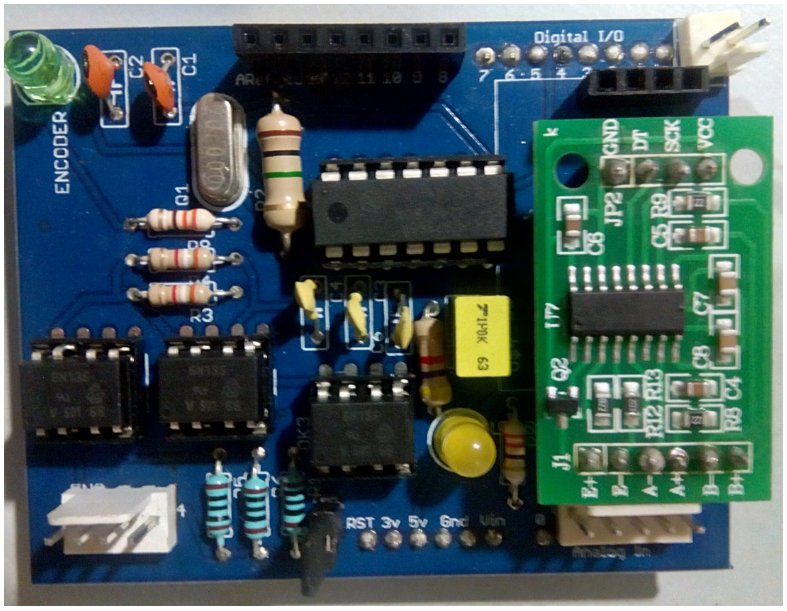

Figura D.3: Fotografia do topo da *shield* desenvolvida depois de soldada e montados os componentes

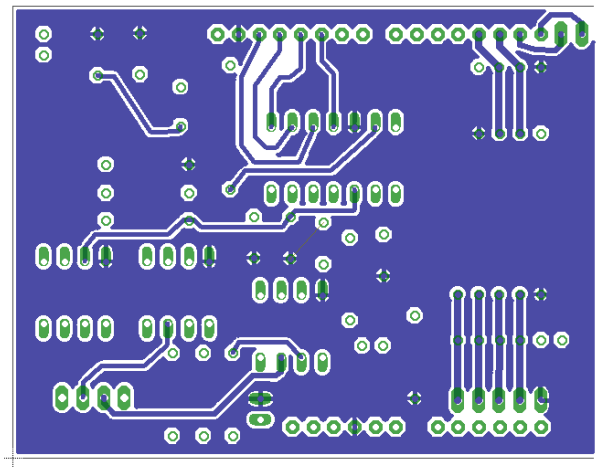

Figura D.4: Vista de base da *shield* desenvolvida

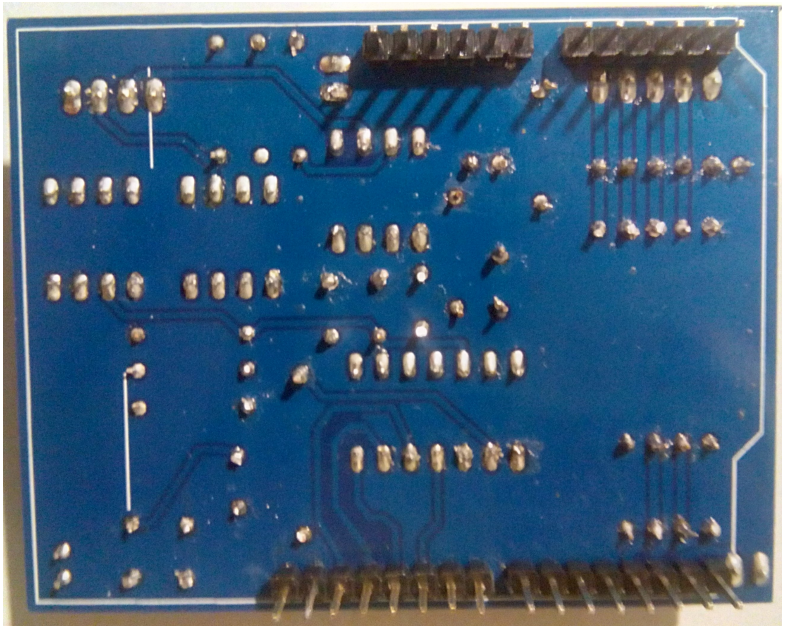

Figura D.5: Fotografia da base da *shield* desenvolvida depois de soldada e montados os componentes